MICROCOMPUTADOR CURSO PRÁTICO

TIRE O MÁXIMO DO MICRO EM 30 SEMANAS

PUBLICAÇÃO SEMANAL ILUSTRADA

16

Cr$ 8.200

Nexus 1684
Gravando na EPROM

Robôs e brinquedos

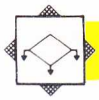
rioGráfica

MANAUS, ALTAMIRA, MACAPA, JI-PARANA, PORTO VELHO, RIO BRANCO, SANTAREM, BOA VISTA, SINOP, ALTA FLORESTA: Cr$ 10.660

MICROCOMPUTADOR CURSO PRÁTICO

APLICAÇÕES

Robôs domésticos 317



LINGUAGENS

Estrutura versátil 320

RAIO X

N1684 325

SOFTWARE

Os nomes dos nomes 328

CHIPS & BYTES

Na velocidade da luz 330

PROJETOS DE PROGRAMAS

Busca de variáveis 332

CIÊNCIA DA COMPUTAÇÃO

Produção de números 334

PERFIL

Unitron 336

TESTE

Depois de ler com atenção todo o fascículo, responda a estas perguntas. Veja a solução no próximo número, junto com um novo teste.

1) Qual a característica fundamental de um robô, que o distingue de um brinquedo?
2) Como se chamam os programas que auxiliam a elaboração e a edição de outros programas?
3) Quais as maneiras de apagar o conteúdo de uma EPROM?
4) Como se chama o visor que produz os dígitos de 0 a 9, utilizado em calculadoras, relógios digitais e alguns micros?

RESPOSTAS DO TESTE ANTERIOR

1) TOTAL é uma variável do tipo real (com parte inteira e fracionária), enquanto TOTAL% é do tipo inteiro.
2) Na escolha de um micro, o principal fator é o software, ou seja, as aplicações a que se destina o equipamento.
3) Os três tipos de sinais que produzem imagens no vídeo de um micro são o de televisão, o de vídeo composto e o RGB (Red, Green, Blue).
4) A folha plástica que se pode colocar sobre um tablete digitalizador (ou outro periférico), contendo comandos específicos para outras aplicações, é a máscara.

PLANO DA OBRA

MICROCOMPUTADOR - CURSO PRÁTICO é uma coleção de 30 fascículos de periodicidade semanal. Os 29 primeiros terão, cada um, vinte páginas internas (miolo) e quatro capas, além de uma introdução de dezesseis páginas acompanhando o fascículo 1. O fascículo 30 incluirá o índice geral da obra. Os miolos, encadernados, constituem dois volumes de um curso prático de microcomputação, com seções indicadas por tarjas de diferentes cores.
As duas primeiras capas são descartáveis; as 3.ª^{as} e 4.ª^{as} trazem programas de jogos. Encadernadas, formam um volume de cem páginas.

COMO ENCADERNAR

As capas duras, incluindo as guardas, estarão à venda simultaneamente com os fascículos 10 (**Volume 1**) e 30 (**Volume 2** e **Jogos**).
Volume 1 - Deve ser encadernado com os elementos dispostos nesta ordem: guardas; frontispício e introdução (publicados com o fascículo 1); miolos dos fascículos de 1 a 15; e guardas.
Volume 2 - Guardas; frontispício (que virá com o fascículo 30); miolos dos fascículos de 16 a 30; e guardas.
Jogos - Guardas; frontispício e sumário (que constituem as 3.ª^{as} e 4.ª^{as} capas do fascículo 30); 3.ª^{as} e 4.ª^{as} capas dos fascículos de 1 a 29; e guardas.

rioGráfica **Diretoria Executiva:** Oscar Neves; Nilo Sérgio de Almeida; Danilo Esteves Costa; **GRUPO EDUCAÇÃO E CULTURA - Planejamento e Comercialização - Gerente de Planejamento:** João Azeredo Coutinho; **Gerente de Marketing:** Jaime Rodrigues; **Gerente de Produto:** José Mauro Porto; **Gerente do P.C.P.:** Ayton Menezes; **Diretor de Vendas:** Marcos March; **Gerente de Vendas:** Rubens Barbosa; **Coordenadores de Promoção:** Paulo Cesar M. Seixas; Edgar Mello M. Neto; **Gerente de Circulação:** Norberto Martin; **Supervisor de Operações:** Abel Pereira Pinto; **Editorial - Diretor:** João J. Noro; **Editor-chefe:** Mauricio Rittner; **Administração Editorial:** Heitor de Souza Paixão; **Editor:** Jesse Navarro; **Revisores:** Isis Augusta Loyolla; Cacilda Guerra; Luiz Vicente Vieira Filho; Marcos Domingos Agathão; Maria da Graça Mendonça Couto; **Diretor de Arte:** Adalberto Cornavaca; **Chefe de Arte:** Bonifácio Duardes Miranda; **Chefe de Estúdio:** José Yuji Kuribayashi; **Assistente de Arte:** Miguel Luis Escámez Simón; **Colaboradores - Editores-assistentes:** Claudio Marcondes; Isa Mara Lando; Luiz Carlos Pizarro Marin; **Texto:** Carlos Eduardo Silveira Matos; Jussara Maturo; Paulo Brito; **Preparação de Texto:** Luiz Carlos Cardoso; **Tradução:** Luiz Roberto de Godoy Vidal; Maria Clara Cescato; Maria Ester Menezes Martinho; Marta Rocha Camargo; Newton Roberval Eichemberg; Regina Amarante; **Pesquisa:** Stella Maria Quentel; **Arte:** Edemilson Ribeiro da Silva; **Assessoria Técnica:** Geraldo Coen; Gustavo José Ferreira Grubba; Lauro de Lauro; Pierluigi Piazzi; Sérgio Jorge Batarce. As fotos não creditadas pertencem à obra original. Seguem-se os créditos das demais: Fernando Scavone; Veja/Editora Abril; Jorge's Estúdio.

© APSIF Copenhagen, 1984; © Orbis Publishing Ltd., 1984; © Editora Rio Gráfica Ltda., 1985, para a língua portuguesa.

Composição: Editora Rio Gráfica Ltda. e AM Produções Gráficas Ltda. **Impressão:** JBIG.

NÚMEROS ATRASADOS - Você poderá comprar os exemplares que faltam em sua coleção, pelo preço do último fascículo posto à venda, em sua banca de jornal preferida. Caso não consiga obtê-los, faça sua solicitação por carta endereçada diretamente à Editora Rio Gráfica Ltda., rua Itapiru, 1209, Rio Comprido, Rio de Janeiro, CEP 20251. O atendimento será feito por via postal. Nas cidades do Rio de Janeiro e de São Paulo, as compras poderão ser efetuadas pessoalmente nos seguintes endereços: Rio de Janeiro - Rua Itapiru, 1209, Rio Comprido; São Paulo - Rua Frei Caneca, 1152, Consolação.

Distribuidor exclusivo para todo o Brasil: Fernando Chinaglia - Rua Teodoro da Silva, 907, Rio de Janeiro. Distribuidor para Portugal: Electroliber Ltda. - Rua Prof. Reinaldo dos Santos, 1488, Lisboa.

ROBÔS DOMÉSTICOS

Da ficção, os robôs já chegaram às lojas. Mas como ter certeza de que se está realmente comprando um? Neste artigo, você tem as respostas para não levar brinquedo por robô. Ou gato por lebre.

Das páginas dos livros, onde surgiram pela primeira vez, os robôs invadiram revistas, jornais e até mesmo filmes. Nas últimas décadas, eles se materializaram nos laboratórios, foram aproveitados na indústria e, hoje, tornaram-se tão triviais que se encontram à venda em lojas, nos países mais desenvolvidos.

O número cada vez maior de pessoas interessadas pela robótica foi o principal responsável pela difusão dos robôs, vendidos prontos ou em forma de kits para montagem. O maior campo de aplicações dessas máquinas encontra-se na educação: para entender os princípios da robótica, nada melhor do que montar robôs e depois programá-los para tarefas úteis (como a soldagem de componentes eletrônicos numa placa de circuito impresso) ou complexas (por exemplo, servir uma xícara de café).

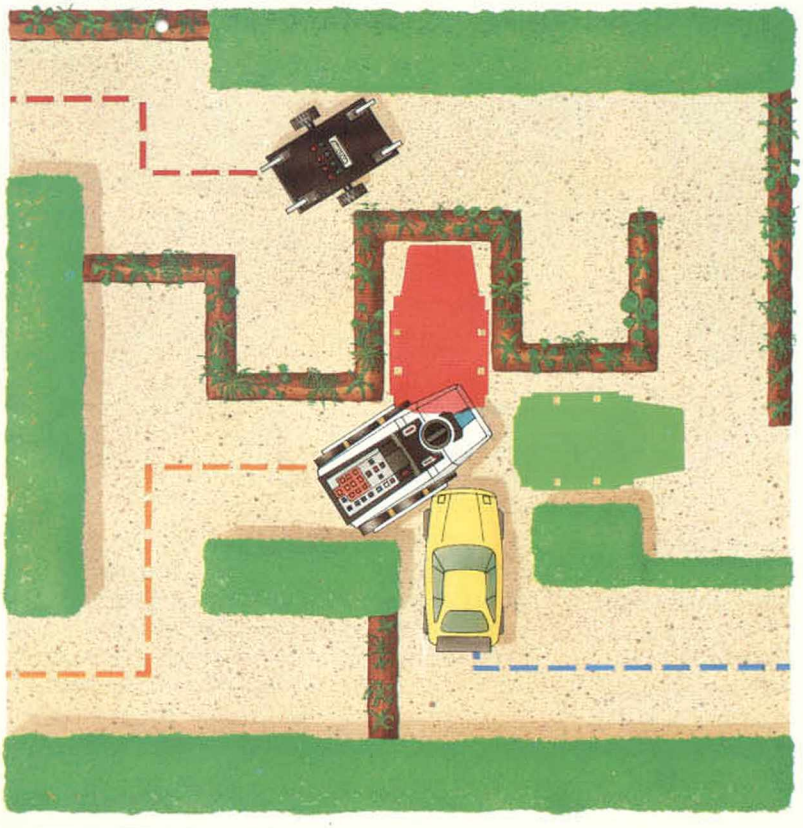
Os robôs que conseguem fazer isso, no entanto, são muito caros: além do hardware eletrônico, exigem componentes mecânicos que incluem eixos, garras especiais, sensores, motores, circuitos hidráulicos e outras peças — sem as quais não seriam máquinas tão versáteis.

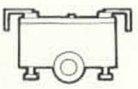
Todavia, quem estiver interessado em compreender o funcionamento dos robôs, não precisa, no início, adquirir um modelo caro e de muitos recursos. Os mais simples, como os mostrados na página seguinte, servem perfeitamente para essa finalidade. Não é incomum que a indústria de brinquedos, no mundo inteiro, anuncie alguns de seus produtos como sendo robôs. Assim, deve-se ter cuidado para não confundi-los com os verdadeiros robôs. Mas como saber as diferenças?

Nem tudo o que se move é robô

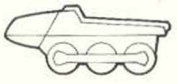
Um dos critérios mais importantes se refere ao movimento. Não se espera que um robô programe a si mesmo, ou que escolha uma seqüência

Descubra o robô





Robô
Controlado por um computador externo, este veículo dispõe de sensores de luz e de pressão.



Big Trak
Segue instruções sobre direção e distância semelhantes às do LOGO. Introduzidas pelo teclado, são programadas em seu próprio microprocessador.

Bumper Car
Equipado com motor a pilha, só anda para a frente. Ao bater num obstáculo vira 90° à direita e continua.

Quem sai dessa?
O teste do labirinto pode ser definitivo para determinar qual dos três dispositivos é um robô. O Bumper Car vira à direita sempre que bate em alguma parede — e pára quando há uma exatamente à sua direita. O Big Trak percorrerá o labirinto com exatidão se as instruções do operador estiverem corretas. Por fim, o robô, com seus sensores, é o único capaz de aprender o traçado, qualquer que ele seja, até sair do labirinto.
Agora, imagine o seguinte: o Bumper Car bate no Big Trak e o desvia 90° de seu curso. Com o choque, o Bumper Car também vira automaticamente à direita; e o Big Trak, sem condições de saber que o curso foi alterado, continua como se nada houvesse acontecido. Só o robô teria uma reação "inteligente", tratando a mudança de rumo como apenas um dos aspectos do ambiente imprevisível pelo qual ele está circulando.

de ações sem orientação humana; mas, uma vez colocado em movimento, ele deverá ser capaz de operar sem necessidade de um controle contínuo. Sem essa liberdade de movimento, um aparelho não é um robô.

Outra avaliação diz respeito ao modo de produção do movimento. Um carrinho de brinquedo com motor a pilhas anda sempre em linha reta. Instale um pára-choques, sensor, mais um dispositivo para virar as rodas dianteiras, e ele se desviará de obstáculos como paredes e móveis. Mude seu centro de gravidade, equipe-o com grandes pneus de borracha e ele poderá subir uma rampa muito inclinada, virar e continuar andando. Embora realize tudo isso sem qualquer controle humano, ninguém vai considerá-lo um robô.

Classifica-se o movimento dos robôs em duas categorias: controlado por um programa e com movimento inteligente. Não se pode programar o carrinho que imaginamos para que se movimente de outro modo; portanto não é um robô. Existem, contudo, brinquedos desse tipo programados por meio de cartões perfurados. Dispositivos mecânicos lêem os cartões e reproduzem os movimentos aí codificados. Nesse caso, estamos diante de um robô.

Ainda há outros aspectos a levar em conta. Por exemplo, a máquina em questão tem sensores para receber informações visuais ou auditivas do mundo exterior? Ela reage a estímulos externos modificando alguma de suas características? Ou — passando para um plano de maior complexidade — ela consegue jogar xadrez? Produz respostas sonoras, simulando a fala humana? A característica essencial, no entanto, para se distinguir entre robôs e brinquedos é o movimento programado.

NOME: Beasty
TIPO: Braço e garra
PROGRAMÁVEL? Sim
SOFTWARE FORNECIDO? Sim
SENSORES: De posição

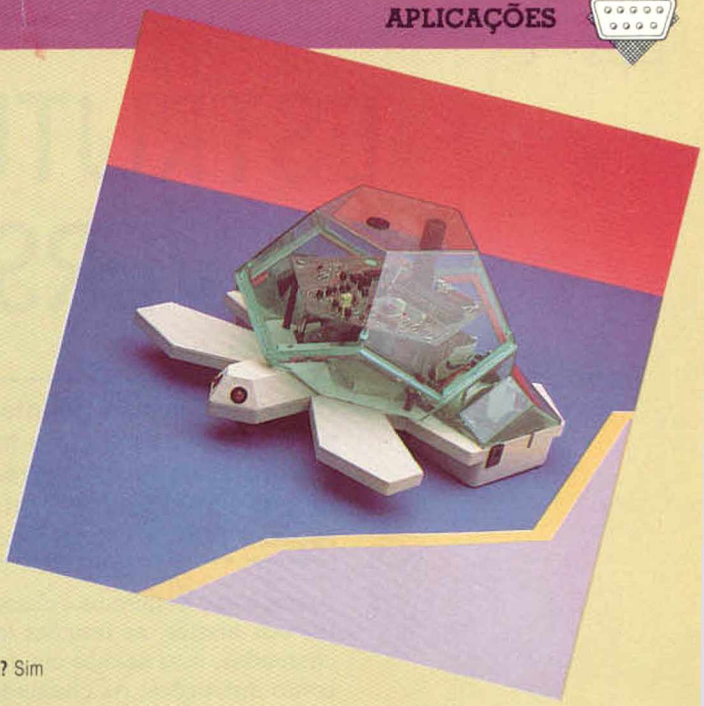
Robô comandado pelo BBC Micro, do qual recebe energia elétrica e sinais de controle por meio do conector de expansões. Utiliza três servomotores. A documentação compõe-se de dois manuais: um referente à montagem do aparelho e outro à sua operação. Ele tem seu próprio sistema operacional, o Robol, por meio do qual se faz a programação dos movimentos (pode-se acionar um motor de cada vez).

NOME: Hebot II
TIPO: Robô de superfície
PROGRAMÁVEL? Sim
SOFTWARE FORNECIDO? Sim
SENSORES: De toque

O Hebot II é um dos tipos mais populares de robôs destinados à educação: projetado para uso com micros da família Sinclair, tem quatro sensores para evitar colisões, um suporte retrátil para caneta (usado em desenhos) e dois motores de corrente contínua, um para cada roda. Segundo o fabricante, embora projetado para o Sinclair, pode ser adaptado para qualquer micro.

NOME: Memocon Crawler
TIPO: Robô de superfície
PROGRAMÁVEL? Sim
SOFTWARE FORNECIDO? Sim
SENSORES: Não

Controlar um robô completamente "cego", como este, pode ser um grande desafio: o Memocon vem equipado com um conjunto de cinco teclas, para que luzes, campainhas e comandos sejam acionados e combinados. Usa dois motores alimentados por pilhas.



NOME: Valiant Turtle
TIPO: Robô de superfície
PROGRAMÁVEL? Sim
SOFTWARE FORNECIDO? Sim
SENSORES: De posição

Robô bastante sofisticado, tem forma de tartaruga, dois motores (um para cada roda), suporte retrátil para caneta e baterias recarregáveis. Comunica-se com o micro por meio de raios infravermelhos.
O software que acompanha a máquina foi desenvolvido na linguagem LOGO, mas isso não impede que funcione com outra linguagem de comando.

NOME: BBC Buggy
TIPO: Robô de superfície
PROGRAMÁVEL? Sim
SOFTWARE FORNECIDO? Sim
SENSORES: De luz e de pressão

Projetado para uso nas escolas inglesas, liga-se ao BBC Micro. Vendido em kits, tem dois motores, alimentação fornecida pela fonte auxiliar do computador, suporte para caneta e, opcionalmente, dispositivo para leitura de código de barras. Com seus sensores, desvia-se de obstáculos e localiza uma fonte de luz.

NOME: Edinburgh Turtle
TIPO: Robô de superfície
PROGRAMÁVEL? Sim
SOFTWARE FORNECIDO? Sim
SENSORES: De posição

Edinburgh é o nome da cidade escocesa onde se desenvolveu esse robô. Ele se liga ao micro por meio de um cabo flexível, pelo qual recebe a energia para acionar seus dois motores (um para cada roda).
Os acessórios incluem um suporte retrátil para caneta (o robô também funciona como plotter), um alto-falante e, opcionalmente, um módulo para controle remoto.

ESTRUTURA VERSÁTIL

O uso de funções no interior dos parâmetros evidencia a integridade estrutural do PASCAL. Veremos aqui como se cria um algoritmo "informal" — recurso útil na organização de arquivos.

Assim como definimos e chamamos procedimentos por nomes, as funções também podem ser chamadas, mas apenas em expressões — nunca como instruções. As chamadas de procedimentos implicam execução de subprogramas responsáveis por algum processo, ao passo que as de funções somente *calculam* valores. Estes podem ser de qualquer tipo simples, real ou escalar, ou um tipo "indicador" (que examinaremos mais adiante). A única diferença no cabeçalho, além do uso da palavra reservada FUNCTION, está na necessidade de especificarmos o identificador de tipo do resultado. Supondo que o PASCAL não predefinisse a função ODD, poderíamos facilmente defini-la nós mesmos:

```
FUNCTION ODD (NUMERO :
  INTEGER) : BOOLEAN;
  BEGIN
    ODD := NUMERO MOD 2 > 0
  END; {ODD}
```

O valor é retornado no identificador da função, devendo então ser atribuído a ele no corpo da função, como mostramos. Nesse sentido, os nomes das funções equivalem a variáveis nunca inicializadas, e seus valores são calculados sempre que surgem numa expressão. No corpo da função, eles aparecem apenas no LHS (Left Hand Side, "lado esquerdo") da atribuição; se formos descuidados (e prolixos), programando a atribuição como

```
IF NUMERO MOD 2 > 0 THEN
  ODD := TRUE
```

(esquecendo a sentença ELSE ODD := FALSE), poderá haver uma chamada de função que produzirá um resultado indefinido.

Tal como acontece com os parâmetros de procedimentos já examinados, transfere-se o valor do parâmetro efetivo para o identificador integer local (NUMERO) a partir do ponto de "execução". Assim,

```
WRITELN (ODD (SQR (N DIV 100)))
```

que apresenta FALSE como resposta, executa quatro operações:

1) calcula a expressão N DIV 100;
2) passa esse valor integer temporário para SQR como um parâmetro efetivo de valor;
3) fornece seu quadrado e passa-o para ODD; e
4) calcula o resultado booleano e transfere-o como parâmetro para o procedimento WRITELN.

Se N tiver valor 17 enquanto se executa a instrução, qual será seu valor depois desta? Essa pergunta deve parecer ridícula, pois não há razão para se alterar o valor de N durante o cálculo de uma função que o inclua como parâmetro. Os resultados das funções dependem apenas do "mundo como ele é" — ou seja, o valor obtido não passa de uma "função" (sic) do(s) valor(es) de seus "argumentos" ou parâmetros. No PASCAL, o mecanismo para transferir apenas o valor das variáveis garante que, mesmo com a modificação local dos parâmetros no corpo da função (por serem cópias locais), os parâmetros efetivos não serão alterados pela chamada.

Dados globais x parâmetros

Por essa mesma razão, embora o PASCAL não proíba, os dados devem ser acessados globalmente. Toda vinculação de dados entre procedimentos e chamadas de funções é controlada por meio de listas de parâmetros — mesmo que os dados estejam no âmbito do subprograma. A única exceção a essa regra geral diz respeito ao acesso de constantes globais. Por sua própria estrutura, elas não correm qualquer perigo no PASCAL, pois é impossível alterar seus valores. No entanto, quando se fornece um valor de constante como parâmetro, ele se tornará uma variável local e, portanto, não confiável.

```
FUNCTION MINUSCULA
  (CARACTERE : CHAR)          : CHAR;
  {RETORNA A MINUSCULA P/ QUALQUER
    MAIUSCULA}
  CONST
      DISTANCIA = 32; {ASCII DE ORD
        ('a') - ORD ('A')}
  BEGIN
  IF CARACTERE IN ['A' .. 'Z']
    THEN
      MINUSCULA := CHR (ORD
        (CARACTERE) + DISTANCIA)
    ELSE
      MINUSCULA := CARACTERE
  END; {MINUSCULA}
```

Ao definirmos procedimentos, por vezes torna-se essencial alterar os valores de seus parâmetros — caso contrário, não executarão a tarefa para a qual foram projetados. Um exemplo é o próprio procedimento READ. De nada serviria que READ (N) desse um valor a um número inteiro local a READ e o valor de N permanecesse inalterado. Em tais casos, precisamos transferir o endereço de um parâmetro de variável (em vez de seu valor) para que o procedimento remeta diretamente a ele (e não a uma cópia local). Esse mecanismo recebe o nome de "passagem por

endereço'' ou ''por referência'' e, de modo geral, deve ser usado apenas com procedimentos.

Obtém-se o fornecimento de parâmetros de variáveis pela inclusão da palavra reservada VAR antes de qualquer item (ou itens) na lista de parâmetros do procedimento. A sintaxe de uma lista de parâmetros assemelha-se à declaração VAR de um bloco; mas, ao invés de VAR aparecer como delimitador, emprega-se a declaração antes de um item a ser VARiado por um procedimento. Por exemplo,

```
PROCEDURE PROCESSO (VAR
  CONTADOR : LISTACONT);
```

Desenvolvimento de aplicações

Como ilustração dessa poderosa técnica, criaremos um programa capaz de ler alguns nomes introduzidos por meio do teclado e, vinculada a cada nome, uma quantia em dinheiro devida pela pessoa. Para simplificar, empregaremos apenas um string por nome. Uma vez testado, o programa poderia abranger outros detalhes, como endereços, números telefônicos etc. ou, ainda, imprimir os nomes em ordem alfabética ou segundo as quantias devidas.

A estrutura natural dos dados implica uma lista de registros com campos de tipo apropriado, que depois poderão ser incluídos com facilidade. Podemos lidar com cada registro como se fosse uma unidade isolada de dados, mas usando qualquer dos campos que escolhermos como uma ''chave'' para a ordenação. Em primeiro lugar, vamos determinar a representação completa dos dados:

```
CONST
  TAMSTRING    = 20;
  TAMLISTA     = 50;
TYPE
  CARDINAL     = 0 .. MAXINT;
  COMPSTRING   = 1 .. TAMSTRING;
  STRING       = PACKED ARRAY
                   [COMPSTRING] OF CHAR;
  DADOS        = RECORD
                   NOME : STRING;
                   DEBITO : CARDINAL
                 END; {DADOS}
  LIMITES      = 1 .. TAMLISTA;
  REGISTROS    = ARRAY [LIMITES]
                   OF DADOS;
```

Essa especificação opera com cinqüenta nomes e supõe (por enquanto) que nenhum excederá vinte caracteres. Contudo, por meio de definições, é possível modificar esses limites.

Todas as subranges e estruturas foram expressas numa definição TYPE por vários motivos:

1) segurança — por exemplo, um índice de tipo LIMITES nunca excede os limites da matriz;
2) depuração de erros — se ocorrer um ''range error'' durante o teste, a mensagem de erro ajudará a localizar o problema;
3) conveniência e eficácia — as variáveis locais têm tipos existentes, evitando duplicação e economizando memória;
4) necessidade — o PASCAL exige que ''indica-

Vantagem positiva

O programa LENDOCARD trabalha com números positivos, lendo os dados como caracteres. Executa-se a conversão para o valor numérico equivalente subtraindo-se o valor do caractere digital do valor do digito 0. Isso vale para qualquer conjunto de caracteres, pois definem-se como contíguos os caracteres 0...9. Note que o programa só possui duas variáveis globais (no ''nível 0''). Os dados para o processamento de números e a função VALOR, que realiza a conversão, são usados apenas por LERCARD e, portanto, declarados e definidos dentro do procedimento. Assim como está, o programa apresenta apenas uma possibilidade de erro — a introdução de um valor cardinal maior do que o MAXINT do compilador invalida a instrução atributiva do loop WHILE.

```
PROGRAM    LENDOCARD    (INPUT, OUTPUT);

TYPE
      CARDINAL   = 0 .. MAXINT;

VAR
      NUMERO     : CARDINAL;
      OK         : BOOLEAN;

      {1111111111111111111111'11111111}

PROCEDURE LERCARD (VAR N    : CARDINAL;
                   VAR OK   : BOOLEAN);
  CONST
      ESPACO     = ' ';
  TYPE
      ORDINAL    = 0 .. 9;
  VAR
      SIMBOLO    : CHAR;
      DIGITOS    : SET OF CHAR;
        {22222222222222222222222}
                 {NIVEL 2}
  FUNCTION VALOR (DIGITO : CHAR) : ORDINAL;
      {RETORNA VALOR NUMERICO DE DIGITO EM
        QUALQUER CONJUNTO DE CARACTERES}
    BEGIN
      VALOR := ORD (DIGITO) - ORD ('0')

    END; {VALOR}
        {22222222222222222222222}
           {DE VOLTA AO NIVEL 1}
  BEGIN {LERCARD}

    REPEAT
      READ (SIMBOLO)

    UNTIL SIMBOLO > ESPACO;

    DIGITOS := ['0' .. '9'];

    OK := SIMBOLO IN DIGITOS;

    IF OK THEN
      BEGIN
        N := 0;

        WHILE SIMBOLO IN DIGITOS DO
          BEGIN {CALCULAR NA BASE 10}
            N := 10 * N + VALOR (SIMBOLO);
            READ (SIMBOLO);
          END {QUALQUER DELIMITADOR DE CARACTERES
            OU 'EOLN'}
      END      {APOS O NUMERO E IGNORADO}

  END; {LERCARD}
      {11111111111111111111111111111111}
BEGIN {LENDOCARD - NIVEL 0 - PROGRAMA PRINCIPAL}

  PAGE (OUTPUT);
  WRITELN;
  WRITELN ('ESTE PROGRAMA LE INTEIROS');
  WRITELN ('POSITIVOS (NA FAIXA 0 .. ',
           MAXINT : 1,')');
  WRITELN ('O ENCERRAMENTO DESTE PROGRAMA');
  WRITELN ('E CONTROLADO ATRAVES DA DETECCAO DE
           ERROS');
  WRITELN;
  WRITE ('ENTRE COM UM NUMERO : ');
  LERCARD (NUMERO, OK);

  WHILE OK DO
    BEGIN
      WRITELN;
      WRITE ('O NUMERO DIGITADO FOI : ');
      WRITELN (NUMERO : 1);
      WRITE ('NUMERO ?');
      LERCARD (NUMERO, OK);
    END;

  WRITELN ('... DETECTADO UM ERRO ...' : 40);

END.
```

dores'' de tipo para parâmetros sejam identificadores, e não ''literais'' (0 .. 155, por exemplo).

Estruturas dinâmicas

A criação desse ''banco de dados'' implica a seguinte modificação: passamos do uso de funções no interior de parâmetros para o desenvolvimento de um algoritmo informal. Além, é claro, do emprego de técnicas precisas de programação para lidar com arquivos de forma organizada — uma exigência fundamental para a elaboração de qualquer banco de dados.

Começaremos pelo exame de algumas variáveis indispensáveis num algoritmo ''informal'':

```
VAR
  LISTA      : LISTAREG;
  TAMANHO : CARDINAL;
BEGIN
  TAMANHO := 0; {TAMANHO ATIVO DA LISTA}
  {READ DADOS NA LISTA, ATUALIZA
    TAMANHO}
  {CLASSIFIQUE TAMANHO ITENS NA ORDEM
    CORRETA}
  {PRINT TAMANHO ITENS DA LISTA}
END.
```

STUB

Sub-rotina que não efetua qualquer processamento; serve para montar o "esqueleto" do programa, evitando o desenvolvimento explícito de todos os detalhes.

Para traduzirmos esse algoritmo informal em algo mais tangível, exprimimos os três estágios principais do processamento como chamadas de procedimentos (com os nomes apropriados) e relacionamos todos os dados necessários sob a forma de uma lista de parâmetros. Depois de fazer isso, podemos montar um esquema de todos os blocos de procedimentos. Por exemplo, a última instrução do programa será

```
PRINT (LISTA : TAMANHO)
```

Todas as informações essenciais são transferidas ao procedimento na lista de valores de dados e no número de elementos; nesse caso, o cabeçalho não precisará de quaisquer VARs:

```
PROCEDURE PRINT (LISTA : LISTAREG;
                 TAMANHO : LIMITES);
```

Poderíamos prosseguir codificando o procedimento até o final, mas é melhor deixá-lo por enquanto como um ''stub'' BEGIN-END e lidar com outros procedimentos de ''nível 1''. Precisamos atribuir um nome para cada um e escolher os itens de dados a serem passados como parâmetros. Além disso, devemos determinar quais procedimentos receberão esses parâmetros e também se algum desses itens será transferido como parâmetro ''VAR'' (endereço).

Todos os métodos de estruturação de dados examinados até agora possuíam tamanho fixo. Essa especificação vem nas definições TYPE, sendo, portanto, conhecida no momento da compilação. O PASCAL dispõe de estruturas de dados avançadas, que variam de tamanho (e mesmo de tipo, dentro de certos limites) durante a execução do programa. O tamanho de uma estrutura avançada somente é limitado pela capacidade física da memória ou pela disponibilidade do armazenamento de reserva, sendo a estrutura avançada mais comum o arquivo seqüencial.

Assim como as matrizes, cada elemento de um arquivo pode ser de qualquer tipo, simples ou estruturado, com a diferença de que não é possível um arquivo de arquivos. À nossa definição anterior de tipo de registro, poderíamos acrescentar as declarações

```
TYPE
  TIPOARQ = FILE OF DADOS;
VAR
  ARQDADOS : TIPOARQ;
```

que permitem a criação e o processamento de arquivos contendo infinitos componentes, cada qual sob a forma de um registro com nome, débito ou qualquer outro campo desejado.

Da mesma forma que READ (SIMBOLO) serve para a leitura de um único CHAR da entrada de arquivo, usamos READ (ARQDADOS, ITEM) ou WRITE (ARQDADOS, ITEM) para manipular toda uma estrutura de registro como um único item de dados. Caso esses arquivos sejam necessários apenas durante a execução do programa, a única exigência consiste em sua abertura, tanto para leitura como para gravação, por meio dos procedimentos RESET e REWRITE, respectivamente. Para torná-los permanentes, relacionamos os identificadores de arquivo na lista de parâmetros do cabeçalho do programa. Por exemplo,

```
PROGRAM DADOS (INPUT, OUTPUT,
  ARQDADOS);
```

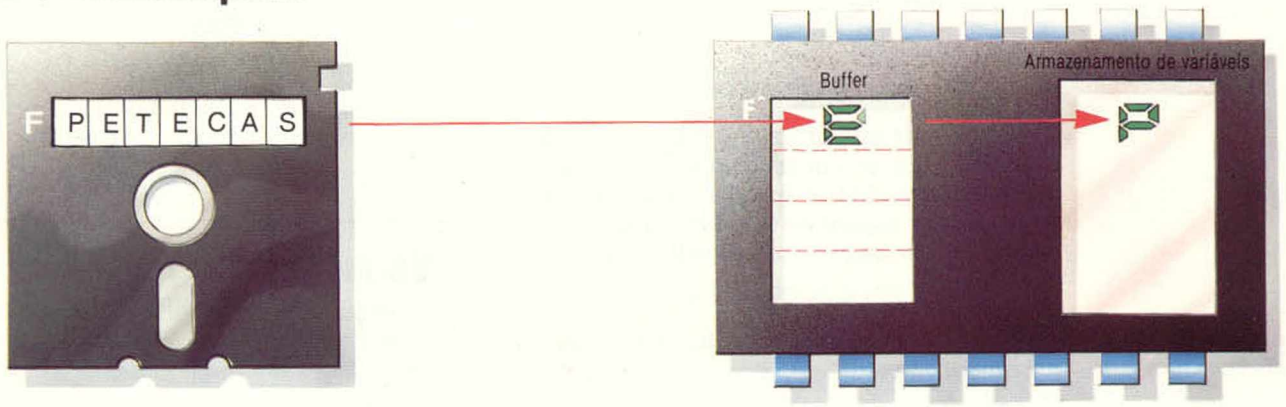
Toda vez que usamos as instruções READ ou WRITE, chamamos os procedimentos de entrada e saída predefinidos do PASCAL. Os dois únicos arquivos que conhecemos até agora são, de fato, os dispositivos padronizados de entrada e saída do computador. Em geral, o arquivo INPUT refere-se ao teclado, enquanto OUTPUT sempre é o vídeo dos sistemas de micro. Ao considerar esses dispositivos como arquivos, toda a manipulação de entrada e saída se torna extremamente coerente no PASCAL.

Lembre-se de que, ao digitar WRITE (N), omitindo qualquer nome de arquivo, o valor de N será impresso sob a forma de caracteres no arquivo OUTPUT. A exclusão de um nome de arquivo em instruções READ ou READLN fará com que seja assumido o default — arquivo INPUT, no caso —, isto é, cada ''registro'' do arquivo será um único valor CHAR.

Arquivos de texto

Os arquivos de texto, ao contrário de outros, possuem, incorporados (além de um marcador de fim de arquivo), marcadores especiais de fim de linha. Estes variam de acordo com o sistema operacional e consistem num único caractere de controle, dois caracteres (CR e LF, por exemplo) ou talvez nenhum, armazenando-se então o tamanho de cada linha.

Leitura de um arquivo



Como garantia de portabilidade, o PASCAL dispõe da função EOLN (F) e dos procedimentos predefinidos READLN (F) e WRITELN (F) — usados apenas com arquivos do tipo texto. Emprega-se a função EOF (F) com qualquer tipo de arquivo. Como o marcador de fim de linha pode ou não ser um único valor CHAR, a leitura desse valor, quando EOLN for verdadeira, fornecerá um caractere Espaço. Assim, devemos sempre verificar EOLN antes. Como os arquivos INPUT e OUTPUT existem independentemente do programa a executar, eles permanecem abertos e não precisam ser localizados ou criados de modo explícito.

No caso de outros arquivos que não INPUT e OUTPUT, temos de atribuí-los a um nome de sistema externo e, então, abri-los para leitura com uma chamada do procedimento RESET — isto é, RESET (UMARQUIVO) — ou criar uma entrada no diretório que prepare a gravação por meio de REWRITE (OUTROARQ). Um esquema geral para o processamento de um arquivo de texto será:

```
{OPEN O ARQUIVO}
WHILE {NAO ATINGIU O EOF} DO
  WHILE {NAO ATINGIU O FIM DA LINHA} DO
    {LE UM CARACTERE}
    {PROCESSA O CARACTERE}
    {SALTA SOBRE O FIM DA LINHA}
```

PUT e GET

Executam-se os procedimentos READ e WRITE com o auxílio de buffers e dos primitivos de entrada e saída PUT (para saída) e GET (para entradas provenientes de arquivos). Quando se abre um arquivo com REWRITE, nenhuma informação entra no buffer até que seja executado um WRITE, o que implica duas operações:

```
F^ := DADOS;
PUT (F)
```

De modo análogo, a instrução READ (F, DADOS) também se exprime por:

```
DADOS := F^;
GET (F)
```

Desse modo, as instruções de entrada e saída em nosso procedimento COPY

```
READ (FONTE, CARACTERE);
WRITE (DESTINO, CARACTERE)
```

também poderiam ser codificadas assim:

```
DESTINO^ := FONTE^;
PUT (DESTINO);
GET (FONTE)
```

Então não precisaríamos requisitar a variável local CHAR, CARACTERE. Talvez seja mais fácil compreender a operação READLN (F) se a expressarmos em termos dos primitivos

```
WHILE NOT EOLN (F) DO
  GET (F); {IGNORA O RESTO DA LINHA, SE
    HOUVER}
GET (F) {E SALTA SOBRE O(S) EOLN(S)}
```

Após um READLN, o buffer do arquivo sempre contém o primeiro item da linha que lerá em seguida. Poderá ser um espaço se EOF (F) for verdadeira. É, portanto, ilegítimo ler um arquivo no caso de EOF ser verdadeira, mas também consistirá num erro realizar qualquer operação, inclusive testes do buffer do arquivo. Por isso, seja cuidadoso ao lidar com outros arquivos que não INPUT e OUTPUT. Essa atenção a condições potenciais de erro garante o desenvolvimento de software extremamente consistente.

Agora que sabemos como "antecipar" um fluxo de dados, passemos a formular o procedimento SALTABRANCOS postulado anteriormente:

```
PROCEDURE SALTABRANCOS (VAR F : TEXT);
  CONST
    ESPACO = ' ';
  VAR
    FEITO : BOOLEAN;
  BEGIN
    FEITO := EOF (F);
    IF NOT FEITO THEN
      FEITO := INPUT^ > ESPACO;
    WHILE NOT FEITO DO
      BEGIN
        GET (F);
```

Na sombra do arquivo
A abertura de um arquivo F no PASCAL implica a criação de uma área buffer correspondente, F^, na qual é lido o primeiro caractere do arquivo. Quando se executa uma operação READ, transfere-se o caractere seguinte do arquivo para o buffer. Se os primeiros caracteres de F forem PETECAS, P será lido em F^. De acordo com a primeira operação READ, P será atribuído a uma variável do PASCAL e substituído por E no buffer F^.

```
      FEITO := EOF (F);
      IF NOT FEITO THEN
        FEITO := F^ > ESPACO
    END
  END; {SALTABRANCOS}
```

Observe que esse programa saltará todos os espaços vazios de um arquivo de texto, inclusive os caracteres de fim de linha. Se queremos apenas saltar o espaço de uma linha qualquer, alteramos a atribuição condicional para

```
FEITO := EOLN (F) OR (F^ > ESPACO)
```

Se, depois, quisermos saltar todos os espaços em branco, faremos

```
REPEAT
  SALTABRANCOS (F);
  IF NOT EOF (F) THEN
    TEMTEXTO := NOT EOLN (F)
UNTIL TEMTEXTO OR EOF (F)
```

Com essa modificação fica fácil desenvolver programas interativos que verificam entradas nulas. Por exemplo,

```
REPEAT
  WRITE ('ENTRE COM DADOS : ');
  SALTABRANCOS (INPUT)
UNTIL NOT EOLN (INPUT)
```

Esse programa será invalidado se introduzirmos o caractere de controle usado pelo sistema para indicar o final do arquivo; mas sempre é possível empregar o esquema anterior para tornar o programa confiável.

Considera-se o procedimento ASSIGN uma "extensão padronizada", e é provável que em breve esteja oficialmente adotado, junto com OPEN e SEEK, para arquivos de acesso aleatório. Outras extensões importantes são

```
FUNCTION FSTAT (FILENAME)
```

que fornece um resultado booleano — verdadeiro, se um arquivo existe — e o recurso

```
PROCEDURE RENAME (OLDNAME, NEWNAME)
```

Variações do PASCAL

No PASCAL, não existem restrições quanto ao número de arquivos que você pode abrir ao mesmo tempo. No entanto, como eles exigem o uso do sistema operacional do computador, essa é uma área onde ocorrem divergências em relação às estritas regras padronizadas. Por exemplo, algumas versões da linguagem não admitem arquivos.

As convenções do sistema operacional relativas à atribuição de nomes e arquivos ocuparão cerca de dez caracteres de qualquer identificador de arquivo. Desse modo, nosso exemplo criaria um arquivo permanente denominado ARQDADOS. Em muitos sistemas, as convenções para atribuição de nomes são tais que isso se torna impossível.

A "extensão padronizada" para vinculação de um identificador de arquivo no PASCAL é o procedimento de atribuição, usado antes de RESET ou REWRITE:

```
ASSIGN (IDENTARQ, NOMESTRING);
```

Assim, para nossos objetivos, ASSIGN (ARQDADOS, 'D : NOSSOARQ.DAD') seria mais do que suficiente.

Programa Copiatexto

O PASCAL dispõe do identificador TEXT para o tipo mais comum de arquivo. Esse identificador é usado na declaração de qualquer variável do arquivo de texto, na lista de parâmetros do cabeçalho, no início do programa. Para tornar o programa mais útil em termos gerais, mostramos o recurso de cópia do arquivo de texto, formulado no procedimento COPY (capaz de processar dois arquivos quaisquer). Os compiladores não padronizados exigirão o uso de RESET (F1, 'FONTE') e assim por diante, no programa principal. Transferem-se a ambos os arquivos como parâmetros VAR para o procedimento COPY. Deve-se isso tanto à desatualização do arquivo a que se destina como ao fato de que o arquivo FONTE será lido e, assim, seu estado se modificará. Por isso, as variáveis de arquivo sempre são transferidas por endereço, nunca por valor. Além de outras considerações, um parâmetro de valor implica uma cópia local — que, junto com os arquivos, sempre ocupa potencialmente mais espaço do que o disponível na memória.

```
PROGRAM     COPIATEXTO     (F1, F2);

VAR
    F1,
    F2     : TEXT;

{11111111111111111111111111111111}

PROCEDURE COPY (VAR FONTE,
                    DESTINO : TEXT);
  VAR
      CARACTERE     : CHAR;
  BEGIN
    WHILE NOT EOF (FONTE) DO
      BEGIN {COPIA UMA LINHA : }
        WHILE NOT EOLN (FONTE) DO
          BEGIN
            READ (FONTE, CARACTERE);
            WRITE (DESTINO, CARACTERE)
          END;
            {COPIA O EOLN}
        READLN (FONTE);
        WRITELN (DESTINO);
      END

  END;     {COPY}
{11111111111111111111111111111111}

BEGIN     {COPIATEXTO - PRINCIPAL}

  ASSIGN (F1, 'FONTE');
  RESET (F1);     {LOCALIZA E ABRE PARA LEITURA}
  ASSIGN (F2, 'DESTINO');
  REWRITE (F2);     {CRIA O ARQUIVO}
  COPY (F1, F2)

END.
```

N1684

Projetado e desenvolvido pela Scopus, o Nexus 1684 presta-se tanto para uso pessoal como profissional. Compatível com o IBM PC, inclui comandos para geração de gráficos, de cor e de som.

A família Nexus 1600 compreende o N1614, o N1615 e o N1684 — microcomputadores projetados e construídos no Brasil, totalmente compatíveis com o IBM PC.

Recebendo o microprocessador 8088 de 16 bits, com opção para co-processador aritmético 8087, a unidade de sistema abriga três placas e um acionador de disquete (no N1614) ou duas (no N1615), elevando a capacidade de 320 para 640 Kbytes. A versão mais recente, o N1684, conta com uma unidade de disquete e uma de disco rígido tipo Winchester de 10 Mbytes, slim (ocupando metade da altura das outras). Essas são as diferenças básicas entre os três modelos.

O teclado do Nexus 1600 é serial, tipo QWERTY, com 86 teclas (dez de função), bloco numérico separado, simulador de ruído mecânico e repetição automática. O design do teclado engloba detalhes que facilitam sua operação: a conexão à unidade de sistema através de cabo espiralado, que aumenta sua mobilidade; e as dimensões reduzidas, leveza e pés retráteis, que conferem maior flexibilidade no uso diário.

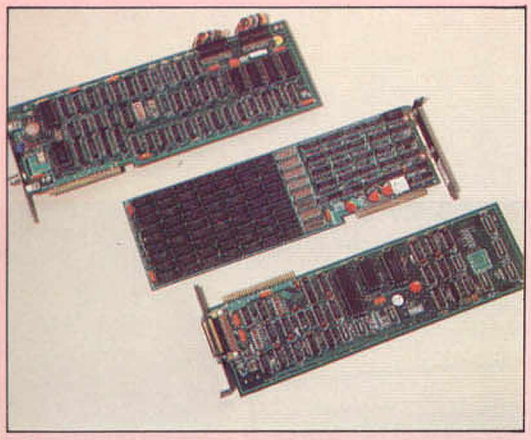
O monitor de vídeo possui cinescópio móvel, adaptável ao ângulo de visão do operador. O Nexus 1600 oferece duas opções de vídeo. Uma delas, o monitor monocromático, de 12 polegadas, em fósforo verde, representa as cores virtuais em tonalidades distintas. É controlado por uma interface gráfica que opera em modo alfanumérico (80 x 25 ou 40 x 25) e gráfico branco e preto (640 x 220 pixels), além de aceitar conexão para caneta óptica. A outra opção consiste num monitor de vídeo colorido, de 14 polegadas, controlado pela mesma placa gráfica já descrita, oferecendo, nesse caso, um modo alfanumérico colorido (dezesseis cores, 80 x 25 ou 40 x 25), ou gráfico colorido (quatro cores, 320 x 200 pixels) e, ainda, o mesmo modo gráfico monocromático (640 x 200 pixels). O monitor dispõe também de máscara anti-reflexo, que elimina cintilações indesejáveis.

Uma placa de funções múltiplas engloba interfaces para joystick, para discos flexíveis de 5 1/4 polegadas e relógio-calendário não volátil, nos dois primeiros modelos. O Nexus 1600 dispõe de mais quatro placas de expansão: a primeira é uma interface para miniwinchester, de 10 Mbytes (específica para o N1614 e o N1615); a segunda, a Incox, para cabo co-axial IBM 327x, possibilita ligação direta com controladoras IBM 3274 e 3276, emulando os terminais IBM 3278 e 3279; as outras são saídas para impressora paralela e expansão de RAM.

Solução integrada
Lançado em 1985, o modelo 1684 da família Nexus já vem equipado com uma unidade de disco do tipo Winchester, com capacidade de 10 Mbytes.

O sistema operacional do Nexus 1600 — o SISNE — foi escrito em PASCAL e, na versão inicial, emula o PC-DOS 1.1 da IBM. Quanto aos aplicativos, o usuário pode utilizar os inúmeros pacotes já existentes no mercado brasileiro. Entre eles, incluem-se os destinados à comunicação de dados: o NX-830, para integração com os sistemas de grande porte da Burroughs; o NX-VIP, para os computadores Honeywell Bull; o NX-Z, de uso geral; e o NX-3270, para IBM.

Assim, interligado em rede com outros computadores, o Nexus 1600 compartilha arquivos e periféricos, reduzindo custos e agilizando operações de processamento. Outra vantagem desse micro está na possibilidade de multiprogramação, que permite executar vários programas simultaneamente.



Solução racional
Além da placa de sistema — que compreende um processador Intel 8088 e duas interfaces de comunicação serial — o Nexus 1684 conta ainda com placas específicas para o controle das unidades de disquete, disco Winchester e monitor de vídeo.

Slots de expansão

Placa controladora de vídeo

Placa de sistema

Unidade de disco flexível

SCOPUS

HÁ 10 ANOS DESENVOLVENDO TECNOLOGIA BRASILEIRA

No Brasil, a era da informática começou recentemente. Mas não para a SCOPUS que, desde 1975, trabalha em projetos, produções, criações e fabricações das mais avançadas, destacando-se no mercado porque aposta no talento e capacidade de técnicos, engenheiros e especialistas brasileiros. Não mede esforços. Sempre à procura de soluções modernas e criativas, desenvolve o que há de mais arrojado em matéria de informática. Sem copiar o que é feito fora. Conheça agora algumas das atuais conquistas tecnológicas da SCOPUS.

Tecnologia e sofistic

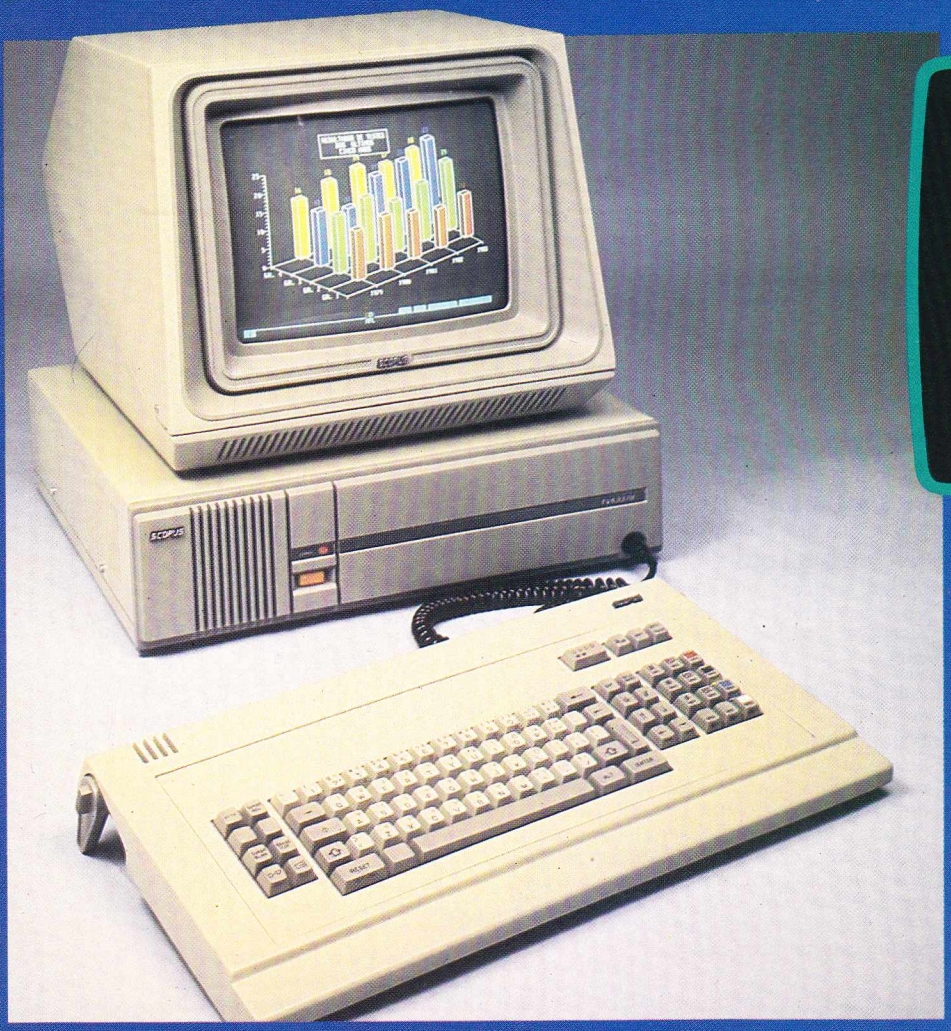
NEXUS 1684

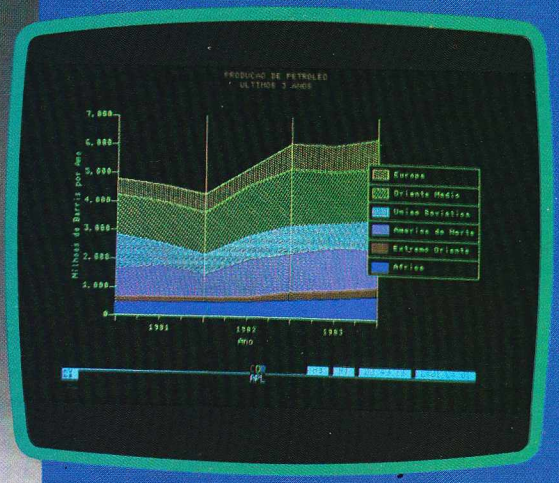
Como o mais recente integrante da família NEXUS 1600 de microcomputadores, o modelo 1684 traz uma importante novidade: a integração da unidade de disco magnético "Winchester", capaz de armazenar 10 milhões de caracteres (10 Mbytes). Dirigido a aplicações profissionais, o NEXUS pode ser utilizado para diversas operações, tais como análise financeira, análise de orçamento, balancete, entre outras tarefas gerenciais e administrativas. Ele se integra também a vários sistemas de grande porte, como computadores IBM, Burroughs ou Honeywell. Oferece características exclusivas, como o "clock" de CPU de 8 MHz e o tratamento de vídeo sem cintilação. Por trabalhar junto com a unidade Winchester, foi possível obter-se uma grande redução de custos além de um melhor aproveitamento do espaço físico.

TVA 3379

Com um hardware sofisticadíssimo e tecnologia (brasileira, é claro) das mais avançadas, o terminal de vídeo TVA 3379 tem como proposta principal facilitar a interpretação de dados fornecidos pelo computador através de gráficos a cores. É o único terminal desse tipo disponível para sistemas IBM existente no mercado. Usando o recurso das cores, o TVA 3379 facilita a leitura e inter-

ção em três versões.





pretação de qualquer informação, o que antes só poderia ser obtido através de tabelas numéricas. É a SCOPUS atenta a todo avanço e tendência do mercado mundial, facilitando a relação entre o homem e a máquina. Apenas usando o talento, o esforço e a criatividade de centenas de engenheiros brasileiros.

PC 2140

O mais novo e econômico "tradutor de linguagens", criado e desenvolvido com tecnologia totalmente nacional, o processador de comunicação PC 2140 permite a conexão simultânea de 32 equipamentos (terminais ou microcomputadores) com um computador de grande porte, através de uma única linha BSC-3. Isto permite o acesso ao banco de dados dos sistemas de grande porte, como IBM ou Burroughs, possibilitando ao usuário levantar importantes informações disponíveis no computador, entre elas, saldos de conta bancária, consultas ao pregão da Bolsa de Valores e verificação do estoque. O PC 2140 pode ser ligado a Unidades de Controle de Transmissão IBM 270X, 370X e ICA. A ligação dos equipamentos ao PC 2140 deve ser direta ou via modem, podendo-se também utilizar o modem de resposta automática.

Qualidade e assistência. A Scopus garante.

A avançada tecnologia do NEXUS 1600, PC 2140 e TVA 3379 já garante ao consumidor alto grau de qualidade e confiabilidade. Mas não é só isso que a SCOPUS oferece. Todos os serviços de manutenção são prestados por equipes de profissionais de alto nível técnico, em constante treinamento, que atendem rapidamente a qualquer chamado. Sempre investindo no aperfeiçoamento de sua estrutura de atendimento, a SCOPUS garante a mais elevada disponibilidade a seus equipamentos e a certeza de uma assistência técnica permanente.

DADOS TÉCNICOS

NEXUS 1600

UNIDADE DE SISTEMA

Placa de sistema: Microprocessador Intel 8088 (4,77 ou 8 MHz) • 64 Kbytes de memória EPROM • 256 Kbytes de memória RAM • 2 interfaces de comunicação serial: assíncrona e síncrona/assíncrona, padrão RS-232C, em velocidades até 19200 bauds • 5 conectores para placas de expansão.

Placa controladora de disco: controla duas unidades de disco flexível de 5 1/4", possui relógio-calendário de tempo real não volátil e duas interfaces para "joy-sticks".

Placa controladora de vídeo: modo alfanumérico: telas de 40x25 ou 80x25 caracteres, 16 cores • modo gráfico: telas de 320x200 pontos a 4 cores ou 640x200 pontos monocromáticas, interface para "light-pen".

Placa controladora de Winchester: controla a unidade de disco rígido tipo Winchester.

Unidade de disco flexível de 5 1/4": um acionador com cabeça dupla, densidade dupla, com capacidade de 320 Kbytes.

Unidade de disco rígido tipo Winchester: um acionador com capacidade de 10 Mbytes, slim.

MONITOR DE VÍDEO

Monocromático, 12", fósforo verde, interface padrão RGB, média resolução e configuração multinível, reproduzindo a informação da cor em tons de intensidades variáveis. Opcionalmente, pode-se configurar o Nexus 1600 com um monitor de vídeo cromático, 14", média resolução.

TECLADO

Tipo máquina de escrever, 86 teclas.

SOFTWARE BÁSICO

O Sistema Operacional SISNE, desenvolvido pela Scopus, emula o sistema MS-DOS 1.1 do microcomputador PC-IBM e contém todos os utilitários compatíveis com os utilitários do MS-DOS 1.1. Possui uma interface homem-máquina bastante poderosa e gerencia, com máxima eficiência, os recursos físicos do equipamento.

MÓDULOS DE EXPANSÃO

N1664: expansão de memória RAM (448 Kbytes) • **N1667**: co-processador CP-8087 • **N1669**: interface paralela para impressora.

MÓDULOS PARA COMUNICAÇÃO DE DADOS

Sistema Incox: interface coaxial para conexão a redes IBM 3270 • **NX-3270**: módulo para comunicação com sistemas IBM sob protocolo BSC-3 • **NX-830**: módulo para comunicação com sistemas Burroughs sob protocolo Poll-Select • **NX-VIP**: módulo para comunicação com sistemas Honeywell Bull sob protocolo VIP 7700 • **NX-Z**: módulo para comunicação assíncrona serial de dados.

TVA 3379

MÓDULOS

Unidade de vídeo: apresenta tela útil de 1920 ou 2560 posições, respectivamente para os modelos 2 ou 3. Possui duas linhas adicionais de informações no vídeo para mensagens do sistema e de erros, além de possibilitar a apresentação de cronômetro e indicador de linha e coluna. O monitor é cromático, de alta resolução. O cinescópio, de 14 polegadas, pode ser configurado para que se apague automaticamente, evitando a queima desnecessária do fósforo.

Unidade lógica: comporta as placas de controle do terminal, de geração de caracteres plano triplo, e as interfaces de comunicação com o vídeo, unidade controladora e teclado.

Teclado: comunica-se com a unidade lógica de forma serial e permite a conexão de um dispositivo auxiliar de entrada. Está disponível em quatro opções: APL Português, APL Inglês, "Typewriter" Português e "Typewriter" Inglês, todas com bloco destacado de teclas de cor e de símbolos programáveis.

PC 2140

CARACTERÍSTICAS FUNCIONAIS

Interface de comunicação com terminais ou micros: serial, assíncrona, código ASCII, full duplex, padrão elétrico RS-232C.

Placa interface de comunicação: cada placa permite que até 8 dispositivos sejam conectados ao PC 2140 com seleção de velocidades independentes.

Conexão de modem: a interface de comunicação com os terminais ou micros permite a ligação de modem, podendo ser constante ou por meio de linhas discadas, através de centrais telefônicas.

Interface de comunicação com o computador: serial, síncrona, sob o protocolo BSC-3, padrão elétrico RS-232C, em velocidades até 9600 bps. Emula as unidades de controle tipo IBM 3271 ou 3274 com 32 terminais IBM 3277 ou 3278, modelo 1.

Conversão de códigos: adotados pelos terminais (ASCII) em códigos adotados pelo computador (EBCDIC).

Execução de comandos: permitem ao computador ativar funções como, por exemplo, estabelecer conexões com os terminais, desfazer conexões, transmitir textos e informar o estado de ligações.

Verificação de paridade: em todas as mensagens com o computador, o PC 2140 gera e verifica os caracteres de controle de bloco (CRC-16), pedindo retransmissão automática em caso de erro.

MODELOS

O PC 2140 está disponível em quatro opções, que diferem apenas em relação ao número de placas de controle dos terminais.

PC 2140/8: controla até 8 terminais (1 placa).

PC 2140/16: controla até 16 terminais (2 placas)

PC 2140/24: controla até 24 terminais (3 placas)

PC 2140/32: controla até 32 terminais (4 placas)

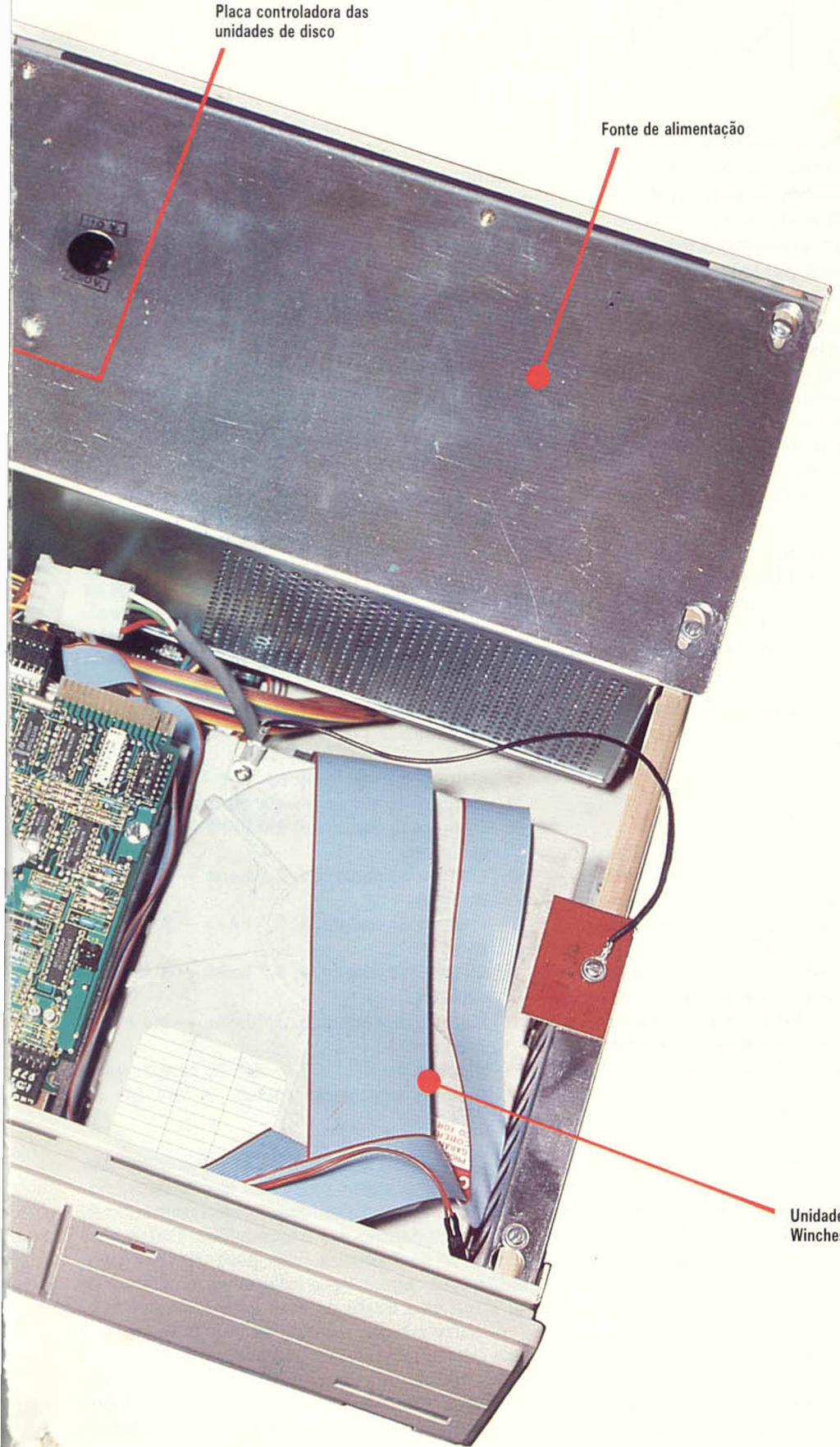
CENTROS DE ASSISTÊNCIA TÉCNICA

BELÉM: Tel.: (091) 228-0011 - ramal 242
BELO HORIZONTE: Tel.: (031) 224-8457
BRASÍLIA: Tels.: (061) 224-9856 e 224-9835
CAMPINAS: Tel.: (0192) 42-7333 - BIP: 657
CUIABÁ: Tels.: (065) 322-0542 321-5792 - BIP: 139
CURITIBA: Tel.: (041) 262-5211 - BIP: 150
FORTALEZA: Tel.: (085) 224-7999 - BIP: DA2
LONDRINA: Tel.: (0432) 22-0558
PORTO ALEGRE: Tel.: (0512) 32-1200 - BIP: 798
RECIFE: Tel.: (081) 221-3100 - BIP: 765
RIBEIRÃO PRETO: Tels.: (016) 636-3967 634-3331/3509 - BIP: 76B
RIO DE JANEIRO: Tels.: (021) 240-4925 e 240-6114
SALVADOR: Tel.: (071) 233-4117 - BIP: D12
SÃO JOSÉ DOS CAMPOS: Tels.: (0123) 22-8247 21-2004 - BIP: 536
SÃO PAULO: Tel.: (011) 255-7744

SCOPUS tecnologia s.a.

Filiada à ABICOMP

As especificações contidas neste folheto podem ser alteradas pelo fabricante sem aviso prévio.



NEXUS 1684

MICROPROCESSADOR

Processador central 8088; co-processador aritmético de ponto flutuante (opcional) 8087.

CLOCK

8 MHz.

MEMÓRIA

256 Kbytes de RAM, expansível até 704 Kbytes; 64 Kbytes de ROM.

VÍDEO

Monitor de 12": monocromático, em fósforo verde, com capacidade multinível (múltiplas tonalidades). Modo alfanumérico, 80 x 25 ou 40 x 25 caracteres; modo gráfico, 640 x 200 pixels. Monitor de 14": em cores. Modo alfanumérico: 640 x 200 pixels, dezesseis cores; modo gráfico: 320 x 200 pixels, quatro cores.

TECLADO

Tipo máquina de escrever com 86 teclas (dez de função), conjunto numérico separado, repetição automática e simulador de ruído mecânico.

LINGUAGEM

BASIC incorporado.

PERIFÉRICOS

Unidades de disco (5 1/4" e miniwinchester), impressora, interface serial RS232C, joystick e caneta óptica.

DOCUMENTAÇÃO

Quatro manuais acompanham o equipamento (apresentação, referência técnica, operação e referência BASIC, fornecendo uma visão das aplicações possíveis do sistema, de sua constituição e das possibilidades de expansão. Além das explicações sobre o modo de operação, informações adicionais orientam a respeito da programação em BASIC.

OS NOMES DOS NOMES

Os gerenciadores de bancos de dados manipulam informações, mas é necessário organizá-las para que o sistema seja útil e eficiente. A importância do planejamento aparece logo nos primeiros estágios.

Para quem usa computadores, a afirmação pode parecer imprópria, mas a praticidade dos livrinhos de endereços como bancos de dados pessoais é impressionante. São até portáteis! Sua única exigência fundamental: que os registros estejam nas páginas destinadas à letra inicial do nome de cada pessoa.

A flexibilidade é tanta que pode haver registros do tipo:

> João Enéias, r. França, 316, Vila Municipal, CEP 13300, Piracicaba, SP, tel.: 83-1511 — tel. consultório do pai (dr. Alfredo): 88-2115, só pela manhã; — à tarde, ligar para 88-2128; muda em jan.

E, mais à frente, uma indicação como esta:

> Vilaça (Maurinho), r. Moura.

Ou, ainda, assim:

> Psiquiatra: 210-6095.

Quem usa um livro de endereços tem liberdade para ordenar os dados da maneira que achar melhor: por nome, sobrenome ou qualquer outro indicador que achar adequado. Isso funciona muito bem no papel, mas, quando se passa um banco de dados como esse para um computador, a abordagem precisa ser melhor sistematizada.

Tomando-se por princípio que um livro de endereços eletrônico terá algumas desvantagens em relação ao original, podemos criá-lo de tal modo que seja pelo menos suficientemente flexível para nos atender em várias situações. Para isso, vamos examinar campo por campo os registros que iremos usar, começando pelo do nome.

Nomes de pessoas têm duas partes: a chamada "metade genérica" (o sobrenome) e a "metade específica" (o nome de batismo). Para certas circunstâncias, convém tomar conhecimento de que, em países como Japão, China e Hungria, o sobrenome vem escrito antes do nome.

Será preciso prever, ainda, a extensão máxima que vai ocupar cada registro. Se o sistema de banco de dados tiver comprimento de campo fixo, imagine todas as situações possíveis — por exemplo, a entrada de sobrenomes de dezoito letras.

Como teremos, de qualquer maneira, de impor um formato, é preferível usar o campo do sobrenome como chave primária de localização do registro.

Depois, vem o problema do endereço. Você já tentou preencher um formulário onde não há espaço para o número do seu apartamento? Pois o problema aqui se assemelha. Se você tiver de lidar com endereços do exterior, a situação pode ficar ainda mais complicada. No Japão, só para citar um exemplo, quando se escreve um endereço, o nome da cidade vem em primeiro lugar. O nome da pessoa é indicado por último.

Especificações básicas

Tanto neste banco de dados que montaremos como exemplo quanto em outros, mais complexos, será praticamente impossível atender a todos os imprevistos. Mas convém admitir pelo menos algumas premissas. Por exemplo: é pouco provável que você tenha de manter nesse arquivo um endereço da Coréia ou do Japão, ou que precise registrar alguém com dezesseis nomes. Então, a estrutura básica poderia ser:

- Campo do sobrenome — até 40 caracteres;
- Campo do nome — até 60 caracteres;
- Campo de endereço, 1.ª linha — até 80 caracteres;
- Campo de endereço, 2.ª linha — até 80 caracteres;
- Campo de endereço, 3.ª linha — até 80 caracteres;
- Campo de endereço, 4.ª linha — até 80 caracteres;
- Campo de endereço, 5.ª linha — até 80 caracteres;
- Campo do telefone — até 20 caracteres; e
- Campo para observações — até 80 caracteres.

Uma especificação básica de registro como essa deve cobrir a maior parte dos imprevistos. Assim, pode haver dificuldades com o tamanho limitado do campo das observações. Outro problema: se você precisa fazer pesquisas por país, então esse campo deve estar separado. Decida-se a respeito ainda na fase inicial do sistema.

O software gerenciador de banco de dados é que determinará a facilidade com que se farão essas mudanças. Este artigo toma como exemplo um gerenciador muito prático, o Card Box (caixa de fichas). O programa tem limitações em

seus recursos de acesso e manipulação de dados, mas, para quem está mais interessado em rapidez sem muita confusão, os resultados satisfazem.

Ele não oferece, como outros gerenciadores, uma linguagem de programação para manipular arquivos ou registros. Em compensação, permite que você recupere registros usando comandos bem simples. Para criar o banco de dados que explicamos a seguir, os comandos também são muito simples.

Usando o Card Box

A primeira coisa a fazer, neste ou em qualquer outro banco de dados, é escolher um formato para os registros. Isso vai determinar que informações deverão ser armazenadas, sua indexação (isto é, que campos consideraremos chaves para a busca dos registros) e como esses registros aparecerão na tela ou nas listagens da impressora.

Se batizarmos nosso banco de dados com o nome de ENDERECOS, o gerenciador imediatamente poderá criar um arquivo de formatos chamado ENDERECOS.FMT. Esse arquivo é editável, ou seja, podemos modificar a maneira como as informações se apresentam — e até criar outros arquivos de formatos.

Como a maioria dos programas desse tipo, o Card Box permite a inserção de textos permanentes para a descrição de cada campo. Imagine ver no vídeo indicações como estas:

```
06168
3995
86
185.000
bico fino
```

Elas ficariam muito mais claras se os campos estivessem identificados deste modo:

```
NUMERO DO FABRICANTE    06168
NOSSO NUMERO            3995
ESTOQUE ATUAL           86
PRECO                   185.000
MODELO                  bico fino
```

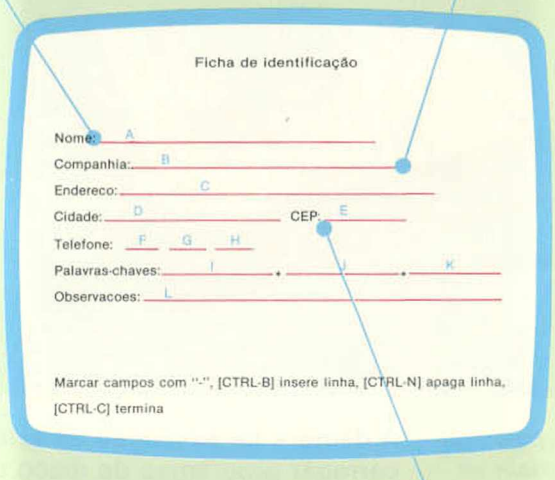
Nos dois casos, as informações são absolutamente as mesmas, mas no segundo minimizou-se a possibilidade de confusão. Para efeito de pesquisa, o programa permite que o usuário dê um destes atributos a cada campo: NADA, MAN(ual), AUTO e TODOS. Num banco de dados de estoque, por exemplo, dificilmente alguém precisaria pesquisar por PRECO. Será bem mais provável que um almoxarife precise saber quantos artigos de número 06168 ainda estão disponíveis para venda. Assim, o campo NUMERO DO FABRICANTE pode ser chave.

Se estivermos usando um banco de dados que contenha os registros sobre empresas e pessoas, precisaremos de buscas por nome, sobrenome, cidade e, às vezes, até pelo código telefônico.

O mais importante de tudo é entender que não se pode criar um bom sistema de banco de dados enquanto não se souber exatamente de que maneira ele será usado. Imagine que você mandasse imprimir fichas de cartolina para arquivá-las numa caixa e ter nelas seus registros. É mais ou menos a mesma coisa quando você projeta seu banco de dados: se não fizer uma previsão correta de suas necessidades, isso poderá lhe custar tempo, dinheiro e aborrecimento.

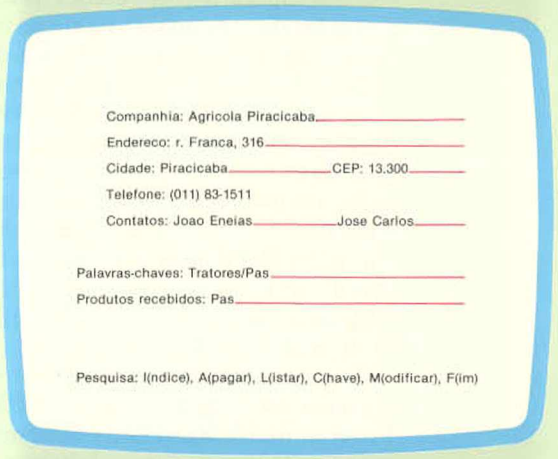
Identificadores
Estes identificadores determinam a ordem de acesso aos registros. Eles tornam possível, por exemplo, a classificação de todos em ordem alfabética, por meio de um comando do programa. Essa classificação pode ser feita por nome, sobrenome, cidade, CEP ou qualquer outro campo escolhido.

Comprimento do campo
Campos grandes facilitam o uso de seu sistema de banco de dados; por outro lado, quanto maiores forem, menos registros poderão ser colocados em cada arquivo. Em geral os campos têm seu comprimento demarcado por traços ou outros caracteres gráficos.

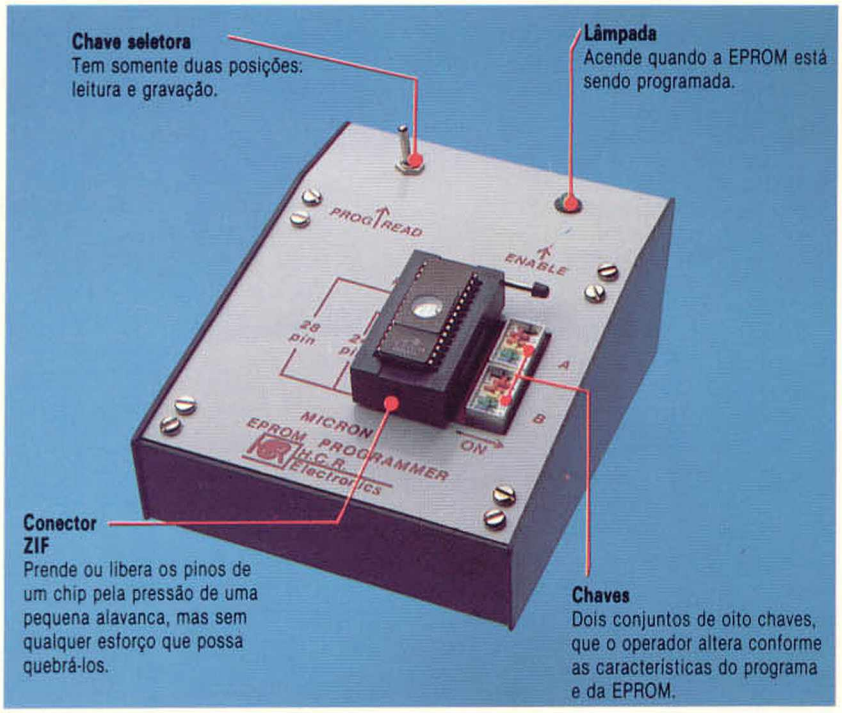


Formato e conteúdo
Acima vemos a definição de um formato para os registros de um banco de dados. A linha inferior mostra os comandos disponíveis nessa fase. A tela abaixo mostra o conteúdo de um registro, com opções para pesquisas e alterações necessárias.

Identificação do campo
Você pode usar os nomes que escolher para identificar cada campo, e faz isso ao projetar o formato do registro.



NA VELOCIDADE DA LUZ



Funcionamento
Um gravador de EPROM insere nos chips programas de uso freqüente. Eles são primeiro carregados na memória do micro e depois transferidos ao gravador.

A demora na execução de tarefas como carregar programas do disco ou da fita pode ser evitada com o gravador de EPROMs — dispositivo de grande utilidade que tende a se popularizar entre os usuários.

Uma das coisas que mais fascinam no primeiro contato com os computadores é sua velocidade: ao simples toque de um tecla, várias coisas acontecem imediatamente. Mas, antes disso, a pessoa que opera o computador precisa executar certas tarefas, as quais, diante da rapidez de alguns programas, parecem lentas demais.

Quando o micro opera com unidades de disco, é necessário ligá-lo, abrir o drive, colocar o disquete, carregar o sistema operacional e, só depois disso, inserir o programa desejado. No caso de equipamentos que operam com fita cassete, o sistema operacional e a linguagem BASIC gravados em ROM estão disponíveis quase de imediato, mas o carregamento de qualquer programa parece uma operação infindável, além de sujeita a erros.

A melhor solução para agilizar esse processo é gravar os programas em EPROM (Erasable Programmable Read Only Memory, "memória programável apenas para leitura e apagável") e utilizá-la do mesmo modo que os cartuchos de videojogos (ou inseri-la nos conectores sobressalentes encontrados em vários micros). Assim, um simples comando faz com que o processador de texto ou outro programa gravado em EPROM seja carregado na memória do micro, a uma velocidade tão alta que parecerá instantânea.

A rigor, é possível gravar em EPROM qualquer espécie de programa. Vários fabricantes mantêm verdadeiras coleções de linguagens e outros utilitários em EPROM para vender a seus clientes: processadores de texto, planilhas de cálculo, linguagem LOGO etc. E você mesmo armazena nelas os programas que usa com maior freqüência.

Como toda ROM, a EPROM é uma densa malha de condutores elétricos, dispostos em linhas e colunas, como uma matriz, onde cada interseção constitui uma "célula". Se numa delas há conexão entre dois condutores, o estado lógico naquele ponto será 1; do contrário, será 0. Quando uma série de impulsos elétricos correspondentes a um endereço chega ao bus de endereçamento do micro, a eletricidade passa em oito desses condutores (os oito impulsos representam 1 byte). Dependendo da existência de conexão nos cruzamentos, a corrente elétrica será reconduzida ao bus de dados do micro. Dessa maneira, transferem-se os dados de determinado endereço para a CPU.

Em geral, os estados lógicos numa EPROM partem todos do valor 1 — cada endereço mantém o valor hexadecimal FF. Mas, se uma corrente de voltagem elevada passa por um célula, a conexão se rompe, criando, dessa forma, o estado lógico 0.

Para gravar 1 byte num endereço qualquer da EPROM, envia-se antes, aos pinos corretos, o conjunto de bits especificador do endereço que se quer gravar. Esse conjunto é transmitido pelo bus de endereçamento. Em seguida, um pulso de 25 V atua por 50 microssegundos sobre os pinos em que se quer estado 0. E o endereço fica gravado. O processo repete-se com os endereços seguintes, até que a EPROM inteira esteja gravada. Em outros tipos programáveis de ROM esse processo não permite reversão, mas uma EPROM é por definição apagável.

A EPROM caracteriza-se por uma pequena janela de quartzo sobre o chip, através da qual se pode vê-lo. Quando uma luz ultravioleta forte atravessa a janela, os átomos dos condutores metálicos que formam a malha se ionizam, e a ligação rompida pela passagem da corrente de 25 V se restabelece.

Isso permite que a corrente volte a circular em todos os cruzamentos da malha de condutores.

Desse modo, o estado lógico torna-se novamente 1 em todos os endereços. A luz ultravioleta não é essencial para apagar a EPROM, já que se pode usar um gravador de EPROM com uma rotina de software para registrar o byte FF (composto apenas por dígitos 1) em todas as posições de memória.

Um bom aparelho desse tipo permite a gravação de até 16 Kbytes de dados e instruções numa EPROM. Os gravadores de EPROM vêm, quase sempre, acondicionados numa pequena caixa metálica, cuja característica que mais se sobressai é um conector ZIF (Zero Insertion Force, "força de inserção zero"). Nele se inserem os chips sem risco de danos para os pinos — que, se quebrados, inutilizam o componente. O ZIF recebe EPROMs de 24 ou 28 pinos. Em alguns modelos, existem dois conjuntos de oito chaves alteradas conforme as características da gravação.

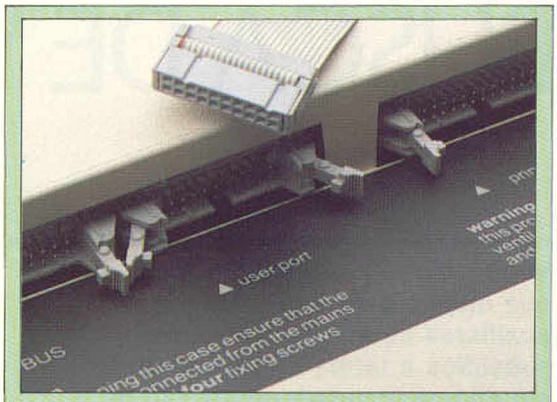
A chave que seleciona as operações de leitura ou gravação da EPROM constitui outro importante dispositivo desses aparelhos. Quando se usa o gravador, é preciso ter certeza de que a chave encontra-se na posição correta; caso contrário, uma EPROM que devia apenas ser lida acaba regravada, o que provocaria o apagamento do programa nela contido.

Em geral, a ligação entre micro e gravador se faz por um cabo flexível multivias, saída serial RS232C ou placa periférica. O programa que vai ser passado para a EPROM fica numa área de buffer da memória do micro, e um software específico auxilia a transferência. Se ele está em cassete, a primeira coisa a fazer é copiá-lo numa EPROM.

Depois de carregados, quase todos os softwares apresentam na tela um menu com as opções disponíveis. Por exemplo: para copiar em EPROM, carregue o programa a ser gravado utilizando a opção L (de load). Esse comando o coloca na área de buffer criada a partir de um endereço determinável. Depois, selecione a opção P (de programas) do menu principal. A tela exibe então a lista dos tipos de EPROM programáveis para escolha. Feita a opção, surgem no vídeo as posições em que devem ser colocadas as dezesseis chaves, quando necessário.

Assim que todas estiverem na posição correta, o programa dará instruções para que a chave seletora de leitura/gravação seja colocada na posição "gravar". Em geral, depois, basta apertar qualquer tecla, que a transferência tem início. À medida que cada posição de memória vai sendo gravada, o número de endereço correspondente aparece no vídeo. O tempo gasto no processo depende da quantidade de endereços a gravar: desde uns poucos segundos (em caso de programa curto) até 14 minutos (para uma EPROM de 16 Kbytes).

Os bons programas de gravação têm vários recursos. Um dos mais úteis é o comando que verifica se a gravação de cada endereço está correta e lista na tela os de conteúdo errado. Não havendo erros, a EPROM pode ser utilizada com segurança.



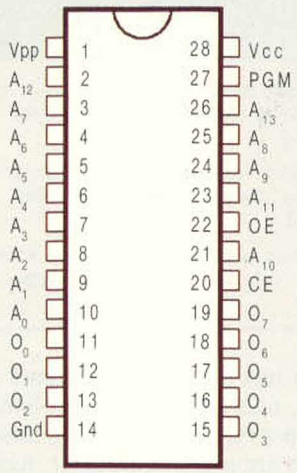
Conexão ao micro
Em geral, por cabo flexível multivias, saída serial RS232C, ou, no caso do Apple, por placa periférica.

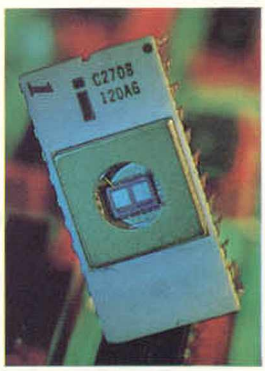
Em vista da grande utilidade desses dispositivos, surpreende verificar que os gravadores de EPROM ainda não sejam populares entre os proprietários de micros. Talvez porque muitos usuários considerem o uso desse aparelho restrito ao "sério" mundo dos profissionais da informática. Puro engano, pois os gravadores de EPROM são fáceis de operar e não exigem qualquer conhecimento de eletrônica. Oferecem, ainda, uma série de vantagens, como a grande velocidade de acesso aos aplicativos e a possibilidade de estes serem chamados pelos programas normais em BASIC.

Pino a pino

Nomes dos pinos	
$A_0 - A_{13}$	Endereços
CE	Habilitação
OE	Saída de habilitação
$O_0 - O_7$	Saídas
PGM	Programa
Vpp Vcc	Alimentação
Gnd	Terra

Diagrama do chip EPROM TMS27128, de 16 Kbytes, com quatorze pinos para endereçamento e oito para saída dos dados armazenados.





A EPROM em close
Pela janela de quartzo podem-se ver os delgados filamentos que ligam o chip aos pinos de seu invólucro e a malha de condutores metálicos (matriz de memória) onde a informação fica registrada.

BUSCA DE VARIÁVEIS

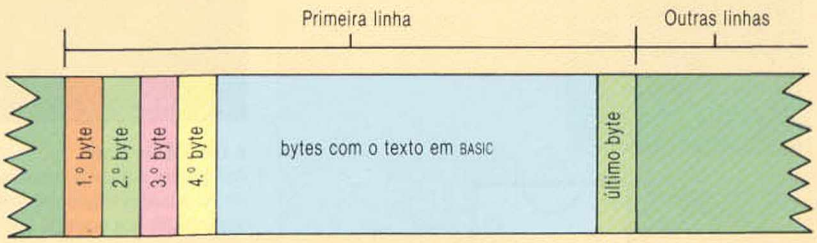
Você mesmo pode criar programas que dispõem de recursos auxiliares de edição ou outros subsídios à tarefa de programação em microcomputadores — são os chamados "utilitários".

Os programas utilitários que vêm incorporados ao BASIC dos microcomputadores variam consideravelmente de fabricante para fabricante. Alguns micros, como TK 82C, TK 83, TK 85, CP 200, Ringo, TK 90X, Unitron, Maxxi, TK 2000 etc., possuem um simples editor de linhas. Outros, como os da linha TRS-80 (CP 500, Sysdata, CP 400 Color), possuem recursos adicionais, extremamente úteis, como AUTO (para numeração automática das linhas) ou RENUM (para renumerar as linhas de programas já existentes).

Os utilitários são geralmente escritos em código de máquina, pela rapidez que ele proporciona e também por não ser nada fácil um programa em BASIC alterar a si mesmo. No entanto, alguns utilitários bem simples podem ser escritos em BASIC. Desse modo, vamos concentrar a atenção no funcionamento e na operação dos utilitários, sem considerar detalhes mais complicados, como as ações do sistema operacional e o próprio código de máquina.

Nos micros Sinclair e Apple, cada linha de programa começa com 4 bytes: dois dedicados ao número e dois ao comprimento (nos TKs) ou ao endereço do início da próxima linha (nos Apples). Após esses bytes iniciais, vem o texto, em BASIC, codificado com as palavras reservadas, os caracteres especiais, os nomes de variáveis, funções etc. Por fim, existe 1 byte destinado a assinalar o final da linha.

Estrutura de uma linha em BASIC



No TK 85: número da linha = 256 x (1.° byte) + (2.° byte)
comprimento da linha = (3.° byte) + 256 x (4.° byte)
último byte = 118.

No TK 90X: número da linha = 256 x (1.° byte) + (2.° byte)
comprimento da linha = (3.° byte) + 256 x (4.° byte)
último byte = 13.

No Apple: número da linha = (3.° byte) + 256 x (4.° byte)
endereço do 1.° byte da próxima linha = (1.° byte) + 256 x (2.° byte)
último byte = 0.

Embora seja difícil um programa em BASIC se modificar, não há grandes problemas em fazer com que inspecione outro programa em BASIC. O utilitário apresentado a seguir, nas versões para as linhas Sinclair e Apple, com variações para o TK 90X, pesquisa sistematicamente um programa em BASIC procurando o nome de uma variável (ou função, no caso do TK 90X e do Apple). Os números das linhas em que ele se encontra são mostrados no vídeo.

O utilitário começa determinando onde tem início, na memória do micro, o texto do programa em BASIC a ser pesquisado. A seguir, percorre o programa linha por linha, saltando os trechos sem nomes de variáveis e extraindo os que encontra. Finalmente, compara os nomes extraídos do programa com aquele que está sendo procurado.

Quando inicia a pesquisa de uma nova linha do texto em BASIC, o utilitário registra o número desta (que é armazenado em 2 bytes) e seu comprimento (nos Sinclairs) ou o número do byte do início da próxima linha (nos Apples). Para maior clareza, observe nos programas ilustrados as estruturas de armazenamento para essas linhas de microcomputador.

O programa salta as linhas REM e as seqüências de caracteres entre aspas, onde em geral não se encontram nomes de variáveis. Os números armazenados em 5 bytes após comandos GOTO ou GOSUB ou atribuições de variáveis também são ignorados.

Depois disso, o programa segue inspecionando cada linha em busca de nomes de variáveis. E estes têm, necessariamente, como caractere inicial, uma letra. Os demais caracteres, quando existem, são letras ou números. As variáveis alfanuméricas apresentam como último caractere o cifrão ($). As matrizes e as funções (no TK 90X e no Apple) têm seus nomes seguidos por parênteses. Considerando essas condições, o programa distingue funções e variáveis numéricas, alfanuméricas e indexadas.

No caso do TK 90X, há outras complicações. As variáveis podem ter nomes em letras maiúsculas, minúsculas ou ambas. Desse modo, as variáveis de nome ALEPH, aleph e Aleph são tratadas como uma só. Portanto, a versão para o TK 90X converte todas as minúsculas em maiúsculas antes de fazer a comparação. O programa para TKs não faz distinção entre matrizes e variáveis alfanuméricas simples. Desse modo, uma matriz T$(4) e o quarto caractere de um string (T$(4)) são considerados a mesma coisa. Mas isso não constitui uma limitação real, pois o BASIC desse equipamento não permite uma matriz e uma variável alfanumérica simples identificadas pelo mesmo nome.

O utilitário deve ser inicialmente digitado e gravado; depois, carregado junto com o programa que se quer pesquisar. O TK 85 e o Apple

necessitam de outros programas auxiliares (os MERGEs), existentes em várias versões, que juntam dois ou mais programas num só.

O TK 90X já incorpora um comando MERGE, bastando carregar o utilitário e aplicar esse comando e o nome do programa a ser pesquisado.

Nos TKs, deve-se comandar RUN 9700 para a execução e, nos micros da linha Apple, GOTO 30000. O utilitário pedirá o nome da variável que deve procurar. Se este for o de uma matriz ou função (para o TK 90X), acrescente um parêntese aberto — (— ao final do nome.

Apple e TK 2000

```
30000 INPUT "QUAL VARIAVEL
      DEVO PROCURAR ?";OBJ$
30010 O1=LEN (OBJ$)
30020 EA=2049
30025 REM ACHA END. PROX.
      LINHA
30030 E1= PEEK ( EA )+256 *
      PEEK ( EA+1 )
30035 REM ACHA NUMERO DA LINHA
30040 NL= PEEK ( EA+2 )+256 *
      PEEK ( EA+3 )
30050 IF NL = 30000 THEN END
30060 CEA= EA + 4
30065 REM COMECA A PROCURAR A
      VARIAVEL NA LINHA
      DO PROG.
30070 FOR X = CEA TO E1
30080 P=PEEK (X)
30085 REM SE FOR REM OU DATA
      PULA PARA A PROX. LINHA
30090 IF P=178 OR P=131 THEN
      EA=E1: GOTO 30030
30095 REM SE FOR ASPAS, VAI
      PARA SUB-ROTINA
30100 IF P=34 THEN GOSUB 30230
30105 REM COMECA A COMPARAR
      LETRA POR LETRA P/
      ACHAR VARIAVEL
30110 CO=0
30120 R=1
30130 FOR N = X TO X + O1 -1
30140 IF ASC( MID$(OBJ$,R,1))=
      PEEK (N) THEN CO=CO+1
30150 R=R+1
30160 NEXT N
30165 REM SE ACHAR VARIAVEL
      VAI PARA 30190
30170 IF CO=1 AND PEEK
      (N-1-O1)>64 AND PEEK
      (N-1-O1)<91 AND(N-1-O1)
      <>(EA+3) THEN GOTO 30190
30180 IF CO=O1 THEN GOTO 30280
30185 REM SE NAO,VOLTA E COME-
      CA UMA LETRA A FRENTE
30190 CEA=CEA+1
30195 REM VE SE NAO E' FIM DA
      LINHA
30200 IF CEA=E1-1 THEN EA=E1 :
      GOTO 30030
30210 NEXT X
30220 EA=E1 : GOTO 30030
30225 REM PROCURA PROXIMA
      ASPAS. ENCONTRANDO,VOLTA
30230 X=X+1
30240 P=PEEK (X)
30250 IF P=51 AND X+2=E1 THEN
      POP : GOTO 30030
30260 IF P=34 THEN RETURN
30270 GOTO 30230
30275 REM IMPRIME O NUMERO DA
      LINHA SE POSSIVEL
30280 P=PEEK (N)
30290 IF P>199 AND P<210 THEN
      PRINT NL
30300 IF P=44 OR P=58 OR P=0
      OR P=198 OR P=59 THEN
      PRINT NL
30310 X=N
30320 P=PEEK (X)
30330 IF X= E1 THEN EA=E1 :
      GOTO 30030
30335 REM PROCURA DOIS PONTOS
30340 IF P=58 THEN GOTO 30190
30350 X=X+1 : GOTO 30320
```

TK 85 e CP 200

```
9700 PRINT "ENTRE A VARIAVEL:"
9705 INPUT T$
9710 PRINT T$
9715 LET REM=234
9720 LET ASPAS=11
9725 LET NL=118
9730 LET NUM=126
9735 LET PROG=16509
9740 LET TP=PROG
9745 LET LIN=256*PEEK (TP)+PEEK
(TP+1)
9750 IF LIN>=9000 THEN STOP
9755 LET TP=TP+2
9760 LET CL=PEEK (TP)+256*PEEK
TP+1)
9765 LET TP=TP+2
9770 LET PL=TP+CL
9775 IF PEEK (TP)<>NL THEN GOTO
9790
9780 LET TP=TP+1
9785 GOTO 9745
9790 IF PEEK (TP)<>REM THEN GOTO
 9810
9795 REM ■PULA LINHAS REM
9800 LET TP=PL
9805 GOTO 9745
9810 IF PEEK (TP)<>ASPAS THEN GO
TO 9855
9815 REM ■PULA TUDO ENTRE ASPAS
9820 LET TP=TP+1
9825 IF PEEK (TP)<>NL THEN GOTO
9840
9830 LET TP=TP+1
9835 GOTO 9745
9840 IF PEEK (TP)<>ASPAS THEN GO
TO 9820
9845 LET TP=TP+1
9850 GOTO 9775
9855 IF PEEK (TP)<>NUM THEN GOTO
 9880
9860 REM ■PULA CINCO BYTES COM
          NUMEROS BINARIOS
9865 LET TP=TP+5
9870 GOTO 9775
9875 REM ■1º CARACTERE DO NOME
          DEVE SER UMA LETRA
9880 IF PEEK (TP)<CODE "A" OR PE
EK (TP)>CODE "Z" THEN GOTO 9895
9885 LET C$=CHR$ (PEEK (TP))
9890 GOTO 9905
9895 LET TP=TP+1
9900 GOTO 9775
9905 LET N$=""
9910 REM ■GUARDA NOME DE VARIAVE
L
9915 LET N$=N$+C$
9920 LET TP=TP+1
9925 REM ■VERIFICA SE O NOME ACA
BOU
9930 IF PEEK (TP)<CODE "0" OR PE
EK (TP)>CODE "Z" THEN GOTO 9945
9935 LET C$=CHR$ (PEEK (TP))
9940 GOTO 9915
9945 REM ■TERMINA COM  $  PARA
          VARIAVEIS STRING$
9950 IF PEEK (TP)<>CODE "$" THEN
 GOTO 9970
9955 LET N$=N$+"$"
9960 LET TP=TP+1
9965 GOTO 9990
9970 REM ■TERMINA COM  (  PARA
          MATRIZES
9975 IF PEEK (TP)<>CODE "(" THEN
 GOTO 9990
9980 LET N$=N$+"("
9985 LET TP=TP+1
9990 IF N$=T$ THEN PRINT N$;" NA
 LINHA ";LIN
9995 GOTO 9775
```

TK 90X

Alterações a serem feitas no programa do TK 85 para que ele rode no TK 90X

```
9706 FOR i=1 TO LEN (T$)
9707 IF T$(i)>="a" AND T$(i)<="z
" THEN LET T$(i)=CHR$ (CODE (T$(
i))-32)
9708 NEXT i
```

```
9720 LET ASPAS=34
9725 LET NL=13
9730 LET NUM=14
9735 LET PROG=23635
9740 LET TP=PEEK (PROG)+256*PEEK
 (PROG+1)
```

```
9880 IF PEEK (TP)>=CODE "A" AND
PEEK (TP)<=CODE "Z" THEN LET C$=
CHR$ (PEEK (TP)): GOTO 9905
9885 REM ■SUBSTITUI  MINUSCULAS
          POR MAIUSCULAS
9890 IF PEEK (TP)>=CODE "a" AND
PEEK (TP)<=CODE "z" THEN LET C$=
CHR$ (PEEK (TP)-32): GOTO 9905
```

```
9930 IF PEEK (TP)>=CODE "A" AND
PEEK (TP)<=CODE "Z" THEN LET C$=
CHR$ (PEEK (TP)): GOTO 9915
9935 REM ■SUBSTITUI  MINUSCULAS
          POR MAIUSCULAS
9940 IF PEEK (TP)>=CODE "a" AND
PEEK (TP)<=CODE "z" THEN LET C$=
CHR$ (PEEK (TP)-32): GOTO 9915
9942 REM ■VERIFICA SE O NOME TEM
          NUMEROS
9943 IF PEEK (TP)>=CODE "0" AND
PEEK (TP)<=CODE "9" THEN LET C$=
CHR$ (PEEK (TP)): GOTO 9915
```

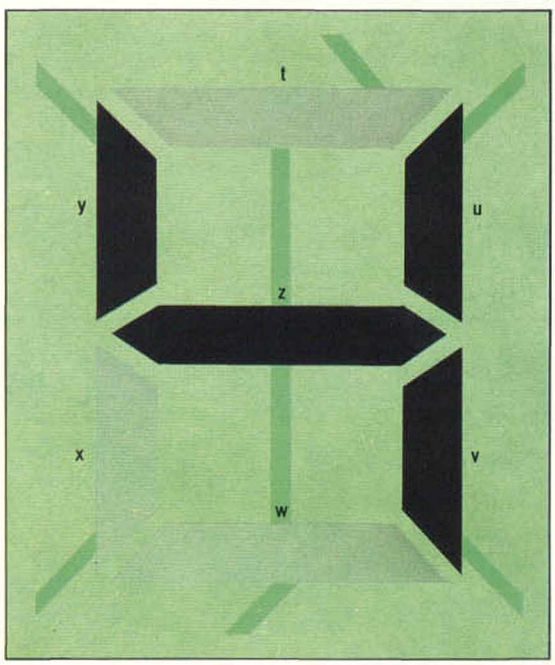
```
9970 REM ■TERMINA COM  (  PARA
          MATRIZES E FUNCOES
```

PRODUÇÃO DE NÚMEROS

Neste artigo do curso de lógica, projetamos um circuito conversor de sinais binários em dígitos decimais, utilizados nos visores de sete segmentos de calculadoras, relógios e alguns microcomputadores.

Em nosso projeto de circuitos, consideremos os nove algarismos decimais codificados em binários (DCB). Isto é, cada dígito decimal é armazenado num grupo de quatro dígitos binários, como mostra a tabela.

Decimal codificado em binário	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010 a 1111	Não utilizado



Visor de cristal líquido
As sete barras do LCD, que podem ser ligadas e desligadas individualmente, aqui designadas por letras de t a z.

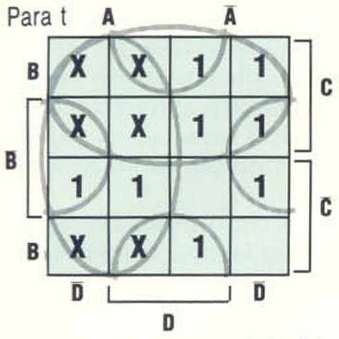
Um visor com sete segmentos (ou barras) representa números decimais por meio de LEDs (Light-Emitting Diodes, "diodos emissores de luz") ou modificando a polaridade de determinadas barras de um LCD (Liquid Crystal Display, "visor de cristal líquido"). A ilustração mostra a denominação dos sete segmentos e as combinações para formar os dígitos de 0 a 9.

A ilustração da página seguinte permite determinar as barras ativadas a cada entrada no circuito. Por exemplo, se a entrada binária for 0100 (4, em notação decimal), o circuito ativará as barras designadas por u, v, y e z. Analogamente, a entrada 1000 (8, em notação decimal) fará com que todas as barras se iluminem. A tabela de validação para o circuito será

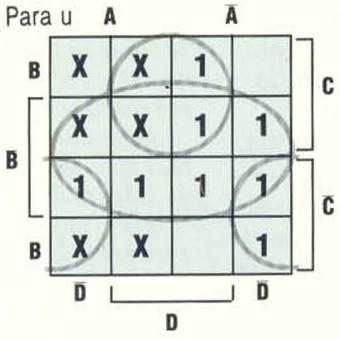
Decimal	Entradas				Saídas						
	A	B	C	D	t	u	v	w	x	y	z
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
	1	0	1	0	X	X	X	X	X	X	X
	1	0	1	1	X	X	X	X	X	X	X
	1	1	0	0	X	X	X	X	X	X	X
	1	1	0	1	X	X	X	X	X	X	X
	1	1	1	0	X	X	X	X	X	X	X
	1	1	1	1	X	X	X	X	X	X	X

Como não há um método fácil de analisar essa tabela, devemos simplificar as saídas separadamente, construindo, para cada uma, um mapa de Karnaugh que inclua as condições "ignore" (X). Em alguns casos, estas oferecem maior simplificação. Os sete mapas-k, um para cada linha, encontram-se na página seguinte. Delimitando os grupos por meio de círculos, produzimos uma expressão simplificada para cada linha de saída. Em alguns casos, fatoramos as expressões para simplificá-las ainda mais.

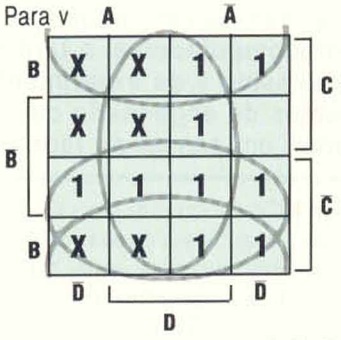
O primeiro mapa-k, por exemplo, lida com as entradas que ativam a barra t (essa barra é usada em todos os números exceto em dois). Podemos agora desenhar o circuito final.



t=C+A+B.D̄+B.D

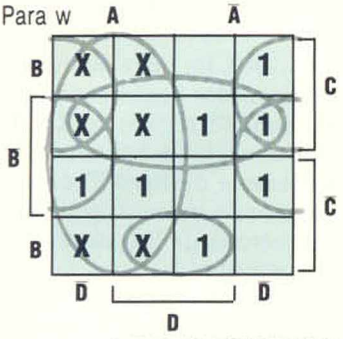


u=B̄+C.D+C̄.D

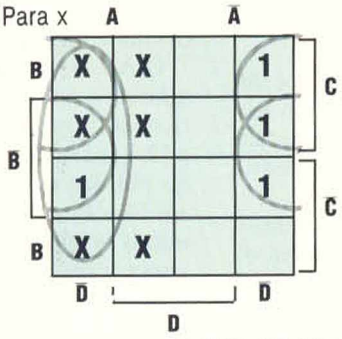


v=D+C̄+B

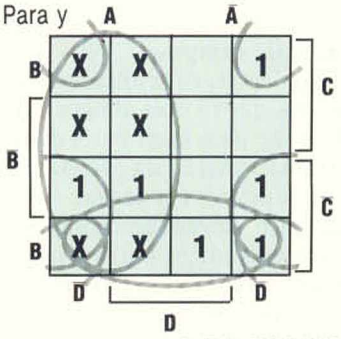
Códigos de barras
A partir destes diagramas determinam-se as barras que devem ser ativadas para formar cada dígito. O algarismo 1 precisa somente das barras designadas por u e v. A tabela de validação na página anterior fornece as saídas necessárias para todos os dígitos.



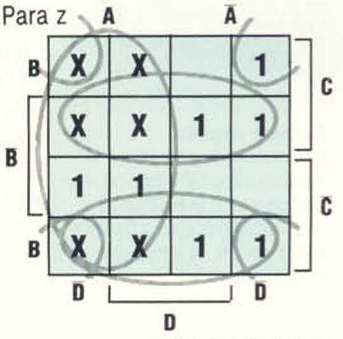
w=A+C.D+B.C+B̄.D+B.C̄.D̄.



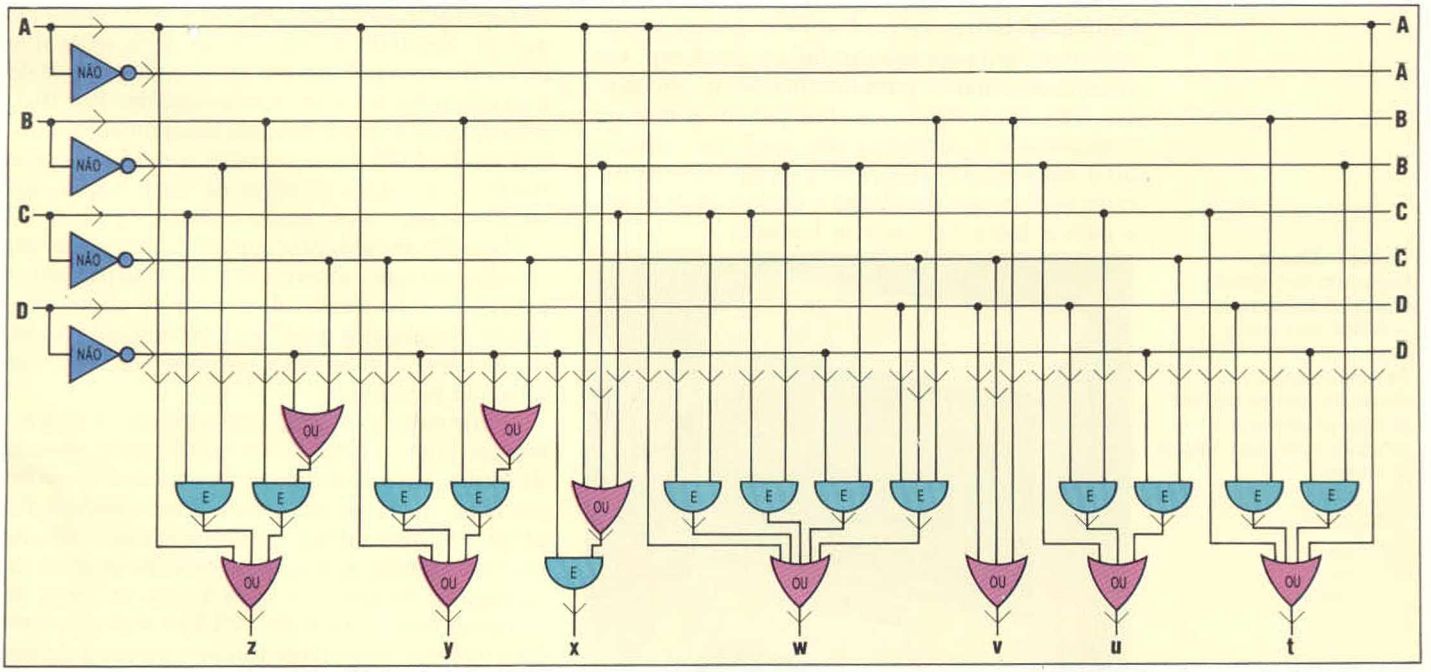
x=C.D+A.D̄+B̄.D
=(A+B̄+C). D



y=A+B.C+B̄.D+C̄.D
=A+B.(C+D)+C̄.D



z=A+B̄.C+B.C̄+B̄.D̄
=A+B̄.C+B̄.(C+D)



O circuito projetado opera uma única célula. Como a maioria dos circuitos desse tipo tem oito ou mesmo dez unidades, poderia parecer necessário duplicá-lo para cada célula. No entanto, é possível partilhar um conversor entre todos os visores, num processo de multiplexação, que liga e desliga as unidades no visor em seqüência — a cada instante, apenas uma unidade admite informações vindas do conversor. Como isso acontece a alta velocidade, o visor parece estável.

UNITRON

Como a maioria das indústrias nacionais de microcomputadores, a Unitron surgiu do entusiasmo e da experimentação de estudantes de engenharia que não hesitaram em passar da teoria à prática.

Os drives elegantes
A Unitron foi a primeira indústria nacional a fabricar um slim drive — o Disk Drive UD 5, uma unidade de discos flexíveis com a metade da altura dos drives comuns.

O ano de 1974 foi crucial para Geraldo Antunes e Ricardo Diez. Nessa época, os dois colegas de faculdade inauguraram uma fabriqueta onde produziriam um aparelho desenvolvido no curso de engenharia. Tratava-se de um dispositivo para acender lâmpadas fluorescentes, ativado por bateria. A produção inicial logo foi absorvida para uso em carrinhos de cachorro-quente e barracas de camping.

Um ano depois, os resultados revelaram-se tão satisfatórios que os dois amigos convidaram Roberto Petrone para fazer parte da sociedade e dirigir a transformação da oficina numa indústria capaz de produzir em escala.

No entanto, devido ao capital inicial mínimo, não restou aos sócios outra alternativa além da exploração das áreas menos visadas no mercado de componentes eletrônicos. Insistiram na mesma linha de produtos: a nova indústria estreou com o lançamento do Reatron — um conversor de corrente contínua, também para lâmpadas fluorescentes. Em 1976, a Unitron desenvolveu seu segundo produto, o Unilamp, um sistema automático para iluminação de emergência. Seis anos depois, os dois produtos haviam conquistado a confiança dos usuários, chegando a empresa a fornecer seus equipamentos para os metrôs de São Paulo e do Rio de Janeiro, e para a Rede Ferroviária Federal.

Teclado inteligente
Programado com memória especial, o teclado desse micro — dotado dos mesmos recursos do AP II — permite a digitação de maiúsculas e minúsculas, além da cedilha e de todos os acentos do português, como numa máquina de escrever.

Ainda em 1982, a empresa instalou-se em prédio próprio e admitiu dois novos sócios, Sérgio Sá Moreira de Oliveira e Vilmar Gaertner. O primeiro foi o responsável pela introdução de novos sistemas e técnicas industriais; Gaertner teve, como diretor técnico, papel fundamental no projeto mais ambicioso da Unitron: entrar no setor da informática.

Nesse mesmo ano, os cinco sócios fundaram a Unitron Eletrônica para fabricar microcomputadores, dividindo instalações com a Unitron Indústria e Comércio. Até o final de 1984, a nova empresa colocou no mercado cerca de 6.500 microcomputadores, divididos entre os dois produtos básicos, o Unitron AP II e o Unitron T.I. (teclado inteligente).

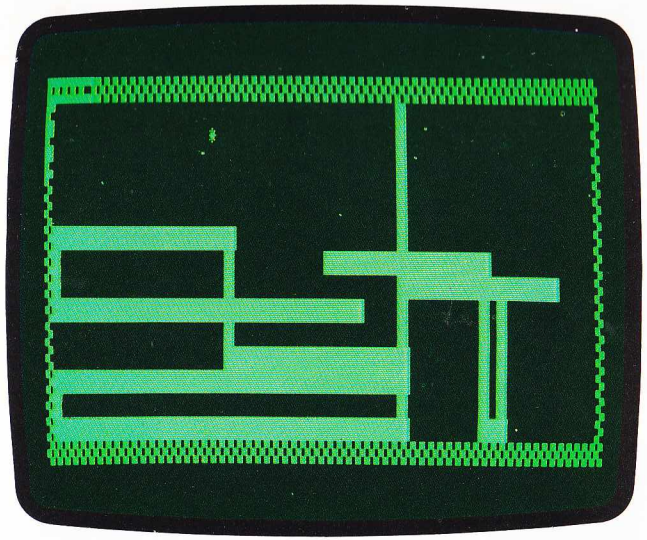
O AP II, lançado em 1982, na Feira de Informática, é um microcomputador pessoal compatível com o Apple II, com 48 Kbytes de RAM e 12 Kbytes de ROM. Já o T.I. foi apresentado ao público em 1984, durante o Micro Festival, em São Paulo. Tratava-se de uma variação do micro original com caracteres da língua portuguesa e teclado programável.

A configuração comum aos dois equipamentos — um monitor de vídeo e unidade de discos flexíveis — possibilita a utilização de impressora e editor de texto. Assim, executam aplicações do tipo folha de pagamento ou de consulta a arquivos. Segundo a Unitron, as aplicações que mais atraem o público em geral estão na área de comunicações e acesso a informações. Por isso, o fabricante instalou em seus equipamentos a interface RS232C, que permite conexão a outro micro ou às redes públicas de informação, como Cirandão, Videotexto e Aruanda.

Uma das grandes preocupações da Unitron diz respeito ao aperfeiçoamento dos equipamentos existentes e ao desenvolvimento de novos produtos. Seguindo a tendência do mercado à verticalização, o fabricante estreou em 1985 sua linha de periféricos.

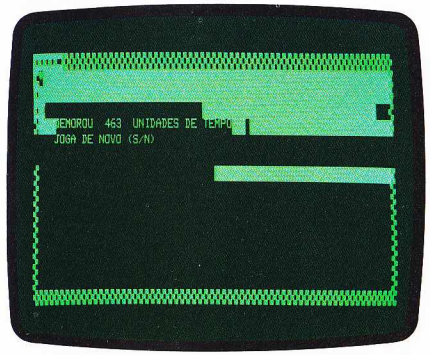
O primeiro aparelho, uma unidade de disquetes tipo slim, o Disk Drive UD 5, tinha metade da altura normal e constituía uma novidade no mercado nacional de drives. Esse produto foi projetado para atingir gradativamente 100% de nacionalização, graças à fabricação própria da cabeça de leitura/gravação e dos motores de acionamento. O Disk Drive UD 5 trabalha com face simples, densidade dupla e capacidade para armazenar até 20 Kbytes.

Como parte de sua estratégia de expansão, a Unitron Eletrônica desenvolveu uma política de comercialização de seus produtos também no exterior, principalmente em países latino-americanos, como a Argentina e o Suriname.



Caminho fechado, jogo perdido
Seu maior cuidado, neste jogo, deve ser o de não fechar o caminho, impedindo a passagem do intruso em direção à armadilha.

```
240 REM
250 REM ***       DESENHA A TELA        ***
260 REM
270 CLS:CLEAR 200:RANDOM:P=15840:PP=191:
    C=15858:CC=42:FOR X=0 TO 4:PRINT
    CHR$(183);:NEXT:FOR X=5 TO 63:PRINT
    CHR$(153);:NEXT:Y=127:FOR T=1 TO 7:
    FOR A=166 TO 153 STEP-13:PRINTà Y,
    CHR$(A);:Y=Y+64:NEXT A:NEXT T
280 FOR X=1023 TO 961 STEP-1:POKE
    X+15360,166:NEXT:Y=960:FOR T=1 TO 6:
    FOR A=166 TO 153 STEP-13:PRINTà Y,
    CHR$(A);:Y=Y-64:NEXT A:NEXT T:FOR
    Y=192 TO 64 STEP-64:PRINTà Y,CHR$
    (183);:NEXT:POKE P,PP:POKE C,CC
290 REM
300 REM ***      VERIFICA TECLADO       ***
310 REM
320 T=T+1:A$=INKEY$:IF A$="" THEN 330
    ELSE N=ASC(A$):IF N=70 THEN 730
    ELSE IF N=90 THEN F=1 ELSE IF N=88
    THEN F=2 ELSE IF N=46 THEN F=3 ELSE
    IF N=44 THEN F=4
330 ON F GOTO 500  ,570  ,620  ,670
340 ON RND(4) GOTO 380  ,400  ,420  ,440
350 REM
360 REM ***  MOVIMENTOS DO INTRUSO    ***
370 REM
380 C=C-64:IF PEEK(C)=191 THEN 390  ELSE
    IF PEEK(C)=183 THEN 720  ELSE IF
    PEEK(C)<>32 THEN 390  ELSE POKE
    C+64,32:POKE C,CC:GOTO 320
390 C=C+64:GOTO 320
400 C=C+64:IF PEEK(C)=191 THEN 410  ELSE
    IF PEEK(C)=183 THEN 720  ELSE IF
    PEEK(C)<>32 THEN 410  ELSE POKE
    C-64,32:POKE C,CC:GOTO 320
410 C=C-64:GOTO 320
420 C=C+1:IF PEEK(C)=191 THEN 430  ELSE
    IF PEEK(C)=183 THEN 720  ELSE IF
    PEEK(C)<>32 THEN 430  ELSE POKE
    C-1,32:POKE C,CC:GOTO 320
430 C=C-1:GOTO 320
440 C=C-1:IF PEEK(C)=191 THEN 450  ELSE
    IF PEEK(C)=183 THEN 720  ELSE IF
    PEEK(C)<>32 THEN 450  ELSE POKE
    C+1,32:POKE C,CC:GOTO 320
450 C=C+1:GOTO 320
```




```
460 REM
470 REM ***  MOVIMENTOS DO JOGADOR    ***
480 REM ***          TECLA  Z          ***
490 REM
500 IF DE<>0 THEN 510  ELSE H=P-64:IF
    PEEK(H)=42 THEN 530  ELSE IF
    PEEK(H)<>32 AND PEEK(H)<>191 THEN
    510  ELSE P=H:POKE P,PP:SU=1:GOTO 340
510 DE=0:SU=0:ES=0:DI=0:F=0:GOTO 340
520 DE=0:SU=0:ES=0:DI=0:GOTO 340
530 J=C-64:IF PEEK(J)=183 THEN 720  ELSE
    IF PEEK(J)<>32 THEN 520  ELSE C=J:
    POKE C,CC:P=H:POKE P,PP:SU=1:GOTO
    340
540 REM
550 REM ***          TECLA  X          ***
560 REM
570 IF SU<>0 THEN 510  ELSE H=P+64:IF
    PEEK(H)=42 THEN 580  ELSE IF
    PEEK(H)<>32 AND PEEK(H)<>191 THEN
    510  ELSE P=H:POKE P,PP:  DE=1:GOTO
    340
580 J=C+64:IF PEEK(J)=183 THEN 720  ELSE
    IF PEEK(J)<>32 THEN 520  ELSE C=J:
    POKE C,CC:P=H:POKE P,PP:DE=1:GOTO
    340
590 REM
600 REM ***      SETA A DIREITA        ***
```



```
610 REM
620 IF ES<>0 THEN 510  ELSE H=P+1:IF
    PEEK(H)=42 THEN 630  ELSE IF
    PEEK(H)<>32 AND PEEK(H)<>191 THEN
    510  ELSE P=H:POKE P,PP:  DI=1:GOTO
    340
630 J=C+1:IF PEEK(J)=183 THEN 720  ELSE
    IF PEEK(J)<>32 THEN 520  ELSE C=J:
    POKE C,CC:P=H:POKE P,PP:DI=1:GOTO
    340
640 REM
650 REM ***       SETA A ESQUERDA      ***
660 REM
670 IF DI<>0 THEN 510  ELSE H=P-1:IF
    PEEK(H)=42 THEN 680  ELSE IF
    PEEK(H)<>32 AND PEEK(H)<>191 THEN
    510  ELSE P=H:POKE P,PP:  ES=1:GOTO
    340
680 J=C-1:IF PEEK(J)=183 THEN 720  ELSE
    IF PEEK(J)<>32 THEN 520  ELSE C=J:
    POKE C,CC:P=H:POKE P,PP:ES=1:GOTO
    340
690 REM
700 REM ***        FIM DO JOGO         ***
710 REM
720 PRINTà 260,"DEMOROU "T" UNIDADES DE
    TEMPO"
730 PRINTà 324,"JOGA DE NOVO (S/N)"
740 G$=INKEY$:IF G$="" THEN 740  ELSE IF
    G$="S" THEN 270  ELSE IF G$="N" THEN
    END ELSE 740
```

GRAND PRIX
PARA LINHA APPLE

Agora você vai mostrar suas habilidades ao volante. No Grand Prix, o desafio consiste em manter o carro na pista sem bater no guard-rail. Pode-se dirigir com paddles ou utilizando o teclado, e escolher entre a veloz Fórmula 1 e a mais lenta das fórmulas V (existem nove níveis de dificuldade). Após digitar [Run], o computador mostrará as opções. Se usar o teclado, você acelera o carro com a barra de espaço e faz curvas com as teclas de setas. Enquanto você permanecer na pista, o computador estará contando seu tempo. Cuidado para não derrapar.

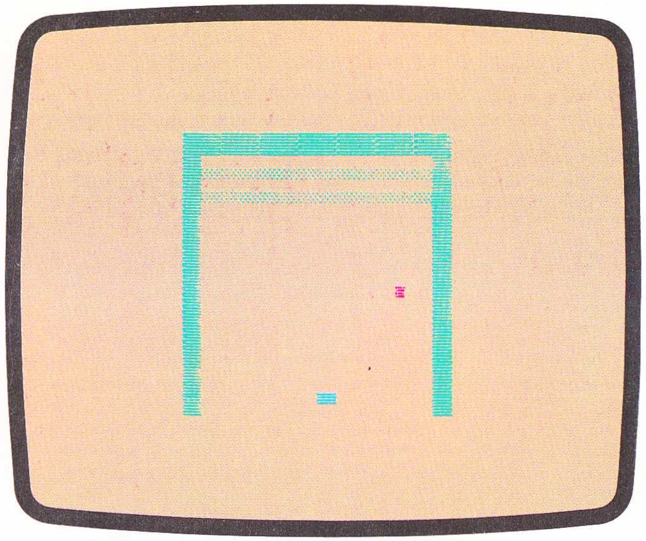
```
10 REM*******************************
12 REM*        PARA LINHA APPLE      *
14 REM*                              *
16 REM*          GRAND PRIX          *
18 REM*******************************
20 REM  **   INICIALIZACAO   **
30 TEXT :HOME :PRINT "PADDLES (S/N):
   ";
40 GET A$:PRINT A$:IF LEFT$(A$,1)="S"
    THEN P=1:GOTO 70
50 IF LEFT$(A$,1)<>"N" THEN 40
60 VTAB 5:PRINT "ACELERADOR -> BARRA
   DE ESPACO":PRINT :PRINT "ESQUERDA
    -> SETA ESQUERDA":PRINT :PRINT "
   DIREITA     -> SETA DIREITA":GOTO 8
   0
70 VTAB 5:PRINT "BUTTON -> ACELERADOR
   "
80 PRINT :PRINT "NIVEL (1-9): ";
90 GET A$:IF A$="" THEN 90
100 PRINT A$:LD=VAL(A$)/18:LU=1-VAL(A$
    )/18:HOME :INVERSE :PRINT "TEMPO";
    :HTAB 25:PRINT "KM"
110 VTAB 24:PRINT "VELOCIDADE";:NORMAL
120 WU=10:WD=5:X%=0:W%=39:C$=" *** ":C
    %=20:M$="!":G$="0 10 20 30 40 50 6
    0 70 80 90 ":DG=2
130 POKE 34,5:POKE 35,22
140 GOTO 400
150 REM  ** DESENHA O CARRO **
160 FOR I=3 TO 5:VTAB I:HTAB C%:PRINT
    C$:NEXT :RETURN
170 REM ** DESENHA A  PISTA **
180 IF G=0 THEN RETURN
190 R=RND(1):X%=X%+(R<LD)*(X%+W%<38)-(
    R>LU)*(X%>1):IF R<LD OR R>LU THEN
    210
200 R=RND(1):W%=W%+(R<LD)*(W%<WU)-(R>L
    U)*(W%>WD)
210 VTAB 22:PRINT :HTAB X%:PRINT M$;SPC(
    W%)M$;:RETURN
220 REM ** INPUT **
230 IF P=1 THEN 310
240 A=PEEK(-16384):IF A<128 THEN 270
250 DC=(DC+(A=149)-(A=136))*(G<>0):IF
    DC>1 THEN DC=1
260 IF DC<-1 THEN DC=-1
270 G=G+(A=160)*DG-(G>0)
280 IF G>27 THEN G=27
290 C%=C%+DC:IF DC=0 THEN POKE -16368,
    0
300 RETURN
310 G=G+(PEEK(-16286)>127)*DG-(G>0):DC
    =((PDL(1)>120)-(PDL(1)<135))*(G<>0
    ):GOTO 280
320 REM ** ACIDENTE ? **
330 IF SCRN(C%,10)+16*SCRN(C%,11)=160 AND
    SCRN(C%+2,10)+16*SCRN(C%+2,11)=160
    THEN RETURN
340 FLASH :PRINT "  ACIDENTE !!!":NORMAL
    :FOR I=1 TO 5:PRINT CHR$(7):NEXT
350 FOR W=1 TO 5000:NEXT :RUN
360 REM ** INFORMACAO **
370 T=T+0.8+0.2*(G<>0):VTAB 1:HTAB 6:PRINT
    INT(T);
```

```
380 M=M+(G<>0):HTAB 31:PRINT M/10"  "
390 INVERSE :VTAB 24:HTAB 7:PRINT LEFT$
    (G$,G+1);:NORMAL :PRINT RIGHT$(G$,
    28-G);:RETURN
400 REM ** PROGRAMA PRINCIPAL **
410 FOR I=1 TO 15:W%=W%-2:X%=X%+1:GOSUB
    210:NEXT
420 GOSUB 170
430 GOSUB 220
440 GOSUB 150
450 GOSUB 320
460 FOR W=0 TO (26-G)*12:NEXT
470 GOSUB 360
480 GOTO 420
```

MINIDEMOLIDOR

PARA LINHA SINCLAIR

Neste jogo, você pode dar vazão a todos os seus impulsos destrutivos.
O programa consiste em demolir uma parede de tijolos. Para tanto, tudo o que você tem a fazer é rebater a pequena bola utilizando a raqueta. Seus controles são as teclas [5] (movimenta a raqueta para a esquerda) e [8] (desloca a raqueta para a direita). Em lugar das teclas, você pode também usar o joystick.



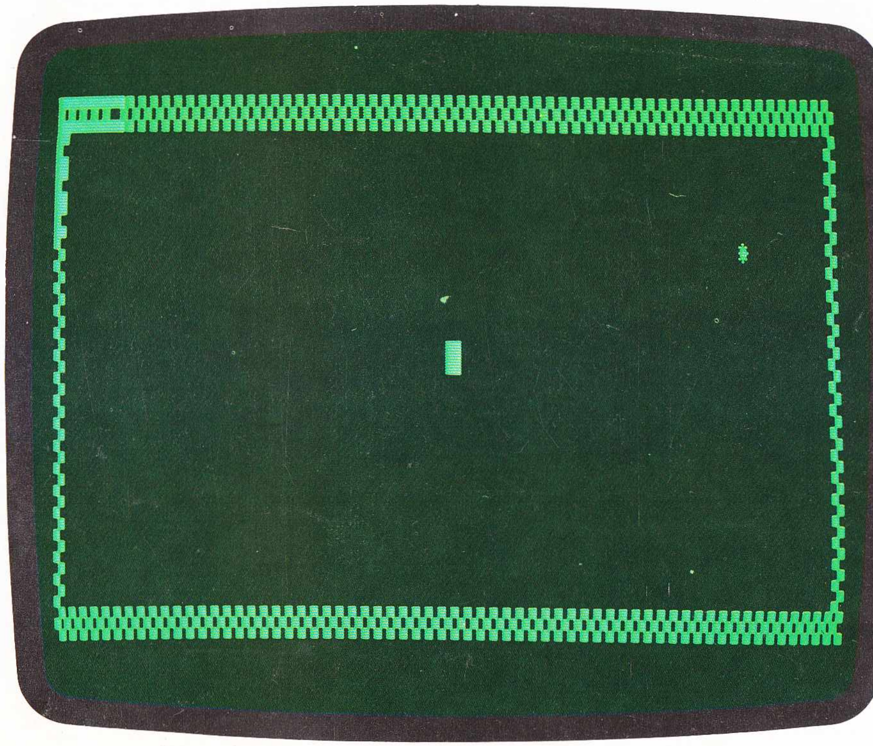
Tijolo por tijolo
Coloque os impulsos destrutivos para fora. Sua tarefa é acertar na bola, rebatendo-a contra a parede. Você soma pontos ao derrubar cada tijolo.

```
 10 REM
 20 REM   MINI-DEMOLIDOR
 30 REM PARA A LINHA SINCLAIR
 40 REM
 50 LET A=13
 60 LET B=A
 70 LET C=21
 80 LET INCX=1
 90 LET INCY=INCX
100 CLS
110 SLOW
120 PRINT "█████████████",,"█
█",,"█                  █"
130 FOR X=3 TO 11
140 PRINT AT X,0;"█
█"
150 NEXT X
160 UNPLOT B,C
170 LET B=B+INCX
180 LET C=C+INCY
190 PLOT B,C
200 IF C>40 THEN LET INCX=INT (
RND*3-1)
210 IF C>40 THEN LET INCY=-INCY
220 IF B<3 OR B>24 THEN LET INC
X=-INCX
230 IF C>21 THEN GOTO 260
240 IF B<>A+1 AND B<>A+2 THEN G
OTO 300
250 LET INCY=-INCY
260 PRINT AT 11,A/2;"  "
270 LET A=A+2*(INKEY$="8" AND A
<23)-2*(INKEY$="5" AND A>2)
280 PRINT AT 11,A/2;"█"
290 GOTO 160
300 PRINT AT 1,1;"FIM DE JOGO"
310 PRINT AT 1,1;" FIM DE JOGO"
320 PRINT ,,"NOVO JOGO (S/N)"
330 IF INKEY$="S" THEN RUN
340 IF INKEY$<>"N" THEN GOTO 30
0
```


INTRUSO

PARA LINHA TRS-80

Seu objetivo aqui é levar o intruso a uma armadilha. Para obter êxito nessa missão, você deve traçar linhas na tela, formando um caminho pelo qual o intruso será empurrado. Ele aparece representado na tela por um asterisco.
As linhas são desenhadas com as teclas [Z] e [X] para os movimentos verticais, e [<] e [>] para os horizontais. Se você fechar o caminho da armadilha, perderá o jogo, mas ainda poderá teclar [F] para determinar o fim e recomeçar.
Quando conseguir encurralar o intruso no canto superior esquerdo da tela, uma mensagem dará o tempo que você gastou para executar a tarefa.



```
100 REM ********************************
110 REM *      PARA LINHA TRS-80       *
120 REM *         INTRUSO              *
130 REM ********************************
140 REM
150 REM ***        INSTRUCOES        ***
160 REM
170 CLS:PRINTà 23," I N T R U S O ";:
    PRINT      "SUA META E' CRIAR UM
    CAMINHO POR ONDE EMPURRARA' O
    INTRUSO (*) ATE' A ARMADILHA DO
    CANTO SUPERIOR ESQUERDO."
180 PRINT "TECLE 'F' PARA INICIAR NOVO
    JOGO."
190       PRINT "PARA CIMA........DIGITE
    (Z)":PRINT "PARA BAIXO.......DIGITE
    (X)"
200 PRINT "PARA DIREITA.....DIGITE (>)"
210 PRINT "PARA ESQUERDA....DIGITE (<)":
    PRINT "PARA PARAR.......DIGITE A
    TECLA OPOSTA";
220       PRINT TAB(19) "QUALQUER TECLA
    PARA JOGAR";
230 IF INKEY$="" THEN 230
```